Extracting and Aggregating Temporal Events from Texts

D I S S E R T A T I O N
zur Erlangung des akademischen Grades

doctor rerum naturalium
(Dr. rer. nat.)

im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Humboldt-Universität zu Berlin

von
Dipl.-Inf. Lars Döhling

Präsidentin der Humboldt-Universität zu Berlin:
Prof. Dr.-Ing. Dr. Sabine Kunst

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät:
Prof. Dr. Elmar Kulke

Gutachter:
1. Prof. Dr. Ulf Leser
2. Prof. Dr. Michael Gertz
3. Prof. Dr. Niels Pinkwart

Tag der mündlichen Prüfung: 22. September 2017

Abstract

Finding reliable information about given events from large and dynamic text collections, such as the web, is a topic of great interest. For instance, rescue teams and insurance companies are interested in concise facts about damages after disasters, which can be found today in web blogs, online newspaper articles, social media, etc. Knowing these facts helps to determine the required scale of relief operations and supports their coordination. However, finding, extracting, and condensing specific facts is a highly complex undertaking: It requires identifying appropriate textual sources and their temporal alignment, recognizing relevant facts within these texts, and aggregating extracted facts into a condensed answer despite inconsistencies, uncertainty, and changes over time. In this thesis, we present and evaluate techniques and solutions for each of these problems, embedded in a four-step framework. Applied methods are pattern matching, natural language processing, and machine learning. We also report the results for two case studies applying our entire framework: gathering data on earthquakes and floods from web documents. Our results show that it is, under certain circumstances, possible to automatically obtain reliable and timely data from the web.

Zusammenfassung

Das Finden von zuverlässigen Informationen über gegebene Ereignisse aus großen und dynamischen Textsammlungen, wie dem Web, ist ein wichtiges Thema. Zum Beispiel sind Rettungsteams und Versicherungsunternehmen an prägnanten Fakten über Schäden nach Katastrophen interessiert, die heutzutage online in Web-Blogs, Zeitungsartikeln, Social Media etc. zu finden sind. Solche Fakten helfen, die erforderlichen Hilfsmaßnahmen zu bestimmen und unterstützen deren Koordination. Allerdings ist das Finden, Extrahieren und Aggregieren nützlicher Informationen ein hochkomplexes Unterfangen: Es erfordert die Ermittlung geeigneter Textquellen und deren zeitliche Einordung, die Extraktion relevanter Fakten in diesen Texten und deren Aggregation zu einer verdichteten Sicht auf die Ereignisse, trotz Inkonsistenzen, vagen Angaben und Veränderungen über die Zeit. In dieser Arbeit präsentieren und evaluieren wir Techniken und Lösungen für jedes dieser Probleme, eingebettet in ein vierstufiges Framework. Die angewandten Methoden beruhen auf Verfahren des Musterabgleichs, der Verarbeitung natürlicher Sprache und des maschinellen Lernens. Zusätzlich berichten wir über die Ergebnisse zweier Fallstudien, basierend auf dem Einsatz des gesamten Frameworks: Die Ermittlung von Daten über Erdbeben und Überschwemmungen aus Webdokumenten. Unsere Ergebnisse zeigen, dass es unter bestimmten Umständen möglich ist, automatisch zuverlässige und zeitgerechte Daten aus dem Internet zu erhalten.

Contents

1 Introduction

Every day, millions of people use the web as an information source, browsing blogs, tweets, newspaper articles, etc. Given long-lasting events developing over time, such as disasters, people seek reliable and timely data describing the event and its aftermath. Understanding "the big picture" in emergency situations, a construct referred to as situational awareness, is obviously essential for effective responses [37]. More and more, information to gain situational awareness can be found in textual or visual form on the internet, both in conventional sources, such as newspapers [29, 120], as well as in social media [57], such as web forums [104] or microblogs [130]. These sources offer among the most detail information available, but searching and analyzing them manually is a time-consuming and therefore costly task.

Today, even rescue teams use the web to gather facts about damages and casualties, especially if no on-site contact is available. Knowing these facts helps them to determine the required scale of relief operations and supports their coordination [117]. Valuable information can be found on the web, but with diverse timeliness, quality, and trustworthiness, imposing several challenges. First, we need to find those documents (web pages, blog entries, tweets) potentially containing the facts of interest (Figure 1.1). Information seeking on the web can by distinguished by two types of interaction: accessing (known) web sites directly or using search engines [55]. Known and relevant sites may be browsed directly, but often we do not know in advance which sources offer the most detailed or up-to-date information. In these cases, invoking search engines are a solution to gather relevant documents, demanding appropriate, i.e. event-relevant queries. Returned search results are heterogeneous documents, probably offering the questioned information encoded as natural language. To gather the desired facts, we have to read and analyze these documents, including assessing their trustworthiness. We furthermore need to align the documents and their contained facts in time. Knowing the temporal dimension of facts is the key to their usefulness, as facts may change over time [56]. Examples are reported results for election polls, which may change from week to week. Since the analyzed documents originate from various sources, published at different points in time, the extracted facts will contain inconsistencies. Resolving these inconsistencies requires adequate aggregation strategies, resulting in condensed views on events. Performing such complex tasks manually is cumbersome, especially if repeated every day on a multitude of sources, clearly calling for automation.

Using the internet as information source carries the danger of "filter bubbles", "echo chambers", "fake news", and "social bots". Echo chamber describes the tendency of internet users to favor those information which are conform with their set opinions and expectations [45]. Consequently, using only known sites to gather information may lead to missing relevant, but potentially non-conform information. Filter bubbles are created by search providers as they tend to personalize search results based on the location of the user and/or the tracked searching and browsing behavior [100]. Such personalized

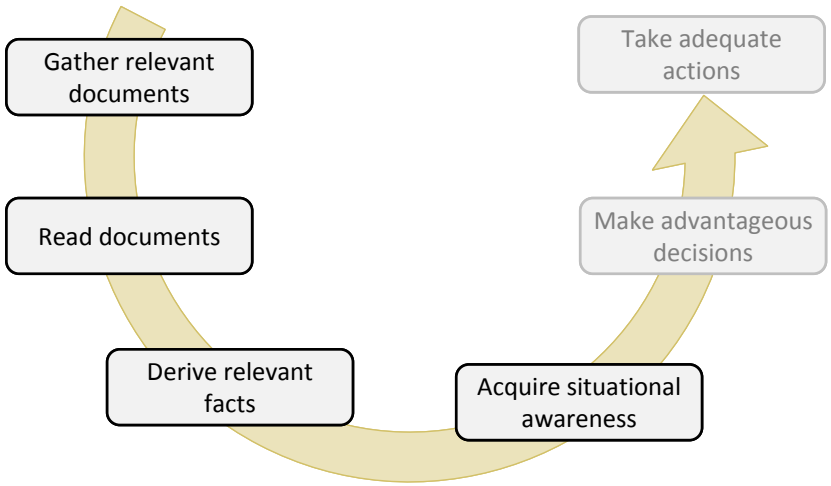


Figure 1.1: Workflow of manual information acquisition to gain situational awareness.

ranking and filtering may lead to hiding potentially relevant documents from users. Fake news are false claims published on the internet, especially on social media. Disintermediated environments, such as social networks, enabled users to generate contents without mediation by journalists or expert, including false contents [28, 131]. Closely connected to fake news are "social bots": computer programs interacting with humans on the same platform, potentially misused to disseminate fake news [41].

Although information extraction from the web has been studied before, to our knowledge, no previous work exists providing comprehensive solutions for all of the outlined challenges at once. There are solutions available targeting specific aspects, but it is unclear, how they perform if combined and given real-world (noisy) input. For instance, Banko et al. created the TextRunner system, capable of efficiently analyzing millions of web pages [9]. By applying a naïve Bayes classifier, they extracted millions of facts from a large collection of web pages and estimated their correctness. Neither inconsistencies between these facts are examined, nor their temporal dimension. Talukdar et al. [123] and Wang et al. [133] examined the temporal scoping of facts in knowledge bases, for example adding the year interval to *IsPresident(Jimmy Carter, USA)*. In contrast to our work, these facts were already known and the temporal scoping was based on already time-stamped texts. Similar to their work, Hoffart et al. created YAGO2, a temporally (and spatially) enhanced knowledge base derived from Wikipedia [56]. Their temporal scoping depends on available meta data, such as the structured infoboxes ('Born = August 8, 1973' or 'In office = May 16, 2010 – October 15, 2012' for politicians) or associated categories ('1999 films' for movies).

Focusing on acquiring situational awareness after natural disasters, we created Equator [32], a web-based content management system meeting specific requirements of the German Task Force Earthquake[1]. It automatically collects, integrates, and visualizes earthquake-related information from the web, reducing the time required to acquire situational awareness significantly. Even though all documents are temporally aligned,

[1] http://www.gfz-potsdam.de/sektion/erdbeben-und-vulkanphysik/themen/mehr/

EQUATOR treads them as "black boxes", still requiring users to read them to obtain the demanded facts. The automatic collection process is also bound to specific sites, neglecting the vast majority of the web. Steinberger et al. created the Europe Media Monitor[2], providing an explorative access to more than thousand news sources in multiple languages [120]. Their system eases manual information browsing by automatic recognition of persons, organizations, etc. as well as categorizing and clustering the documents. They also detect trending topics among their monitored media, tracing their development over time and in space. Still, topic-relevant documents need to be analyzed manually to gather contained key facts.

Based on user-generated content, Sakaki et al. created a system to trace earthquakes and typhoons reported on Twitter, utilizing humans as "social sensors" [113]. They applied particle filters to aggregate tweets into spatiotemporal trajectories describing evolutions of event locations. In contrast to web pages, tweets are already temporally tagged, easing information aggregation.

1.1 Problem Statement

Considering limitations of current approaches leads to our main research questions investigated in this thesis: How can the outlined, complex information acquisition workflow (see Figure 1.1) be automated and what accuracy can we expect?

Given an (exceptional) long-lasting *event*, described by heterogeneous *documents* that are difficult to *find* and a *time*-dependent *information request*: How can we automatically create a reliable view on the event, answering the request? For example, after hurricanes, decision makers require information about where and how many people are injured to coordinate the deployment of medical assistance [54].

In this thesis, we propose a configurable framework providing solutions for the retrieval of relevant documents, the extraction of demanded facts, and their time-resolved aggregation. For each processing step within the framework, users may choose the appropriate resource or method to apply, e.g. what kind of documents to retrieve, which fact extraction method to use, etc. Given an adequately configured framework, the inputs are events—defined by type, date, and location—along with training examples of the demanded facts—representing the information request (Figure 1.2). The outputs are temporally aggregated facts, satisfying the information request by providing condensed views on events.

Events of interest in this thesis are characterized by lasting a substantial period in time, i.e. days or weeks, and having changing information over time. We focus on the temporal aspects of such events, i.e. connecting information with time, and leave their spacial dimension to future work. Furthermore, we focus on English web pages as sources and numerical facts as requested information.

1.2 Contributions & Thesis Outline

Our main contribution is the first proposal, implementation and evaluation of a configurable, all-encompassing framework for solving the complex problem of extracting

[2] http://emm.newsbrief.eu/overview.html

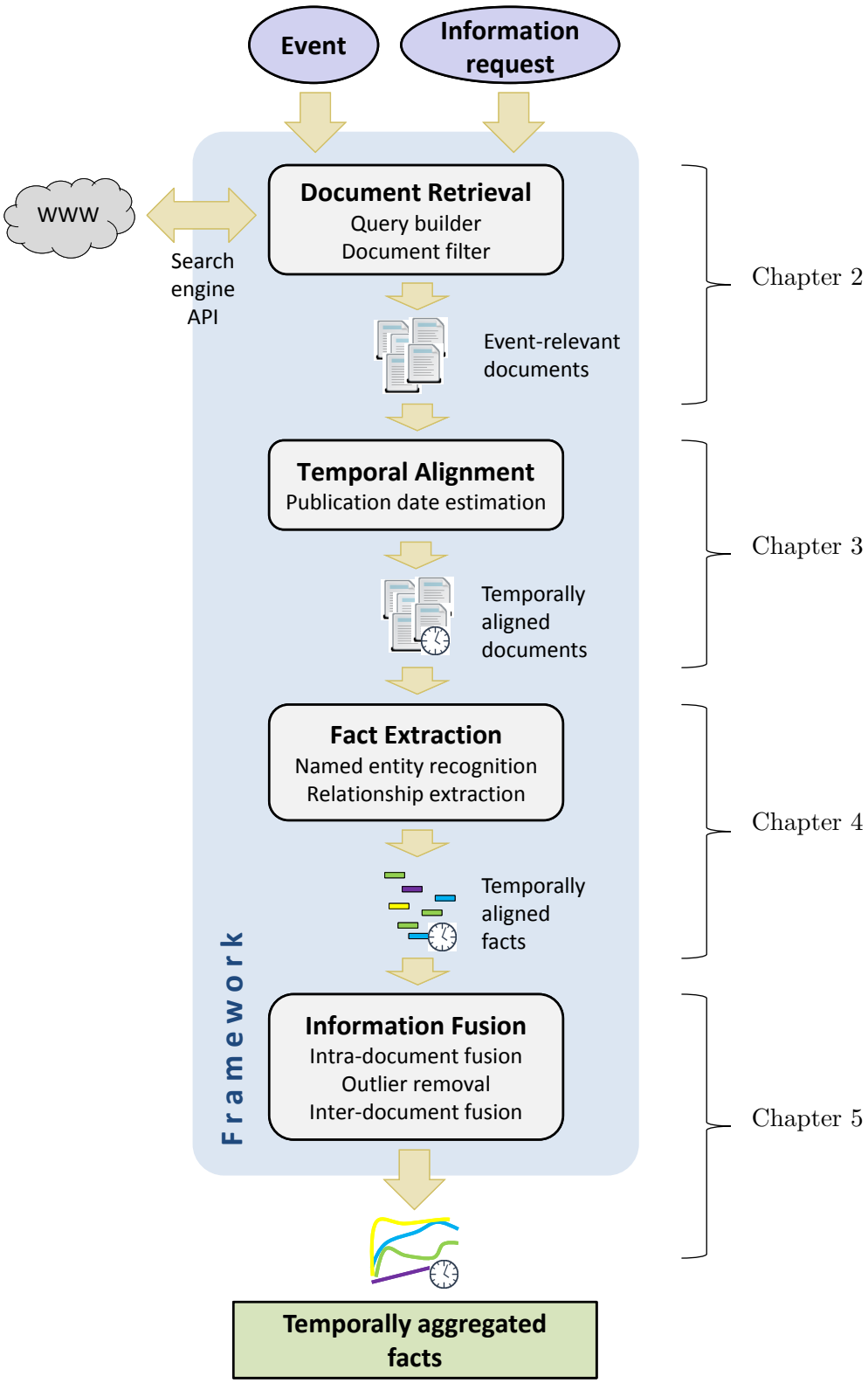


Figure 1.2: Framework overview including references to the respective thesis chapter.

temporal, event-specific facts from web sources. The framework is designed to find, extract, align, and aggregate requested facts from the web describing (long-lasting) events. It especially focuses on the temporal evolution of these facts, providing a reliable, time-resolved view on events despite uncertainty and changes over time. Our framework has four major modules (Figure 1.2), each performing one processing step and described in its respective chapter:

- Document retrieval (Chapter 2),
- Temporal alignment (Chapter 3),
- Fact extraction (Chapter 4), and
- Information fusion (Chapter 5).

Chapter 2 describes a method for retrieving event-relevant documents from the web by invoking search engines, evaluated for an example event. Given an event—defined by type, location, and date—we automatically generate event-specific keyword queries sent to search engine APIs. The module is complemented by document filters removing probably irrelevant search results.

Chapter 3 presents methods for the temporal alignment of (web) documents, evaluated on a novel, specifically designed corpus of web pages. This includes our own rule-based approach for exact temporal alignment of web documents based solely on their content. It uniquely combines three aspects of temporal alignment: providing granularity to the second, identifying last updates, and being independent from external information.

Chapter 4 examines state-of-the-art approaches for extracting facts from documents, i.e. for named entity recognition and n-ary relationship extraction. This includes rule/pattern-based approaches as well as machine learning. We deliver a detailed analysis of their characteristics by evaluating them on three novel corpora consisting of news and Wikipedia articles.

Chapter 5 presents a set of time-aware strategies to deal with inconsistencies in the extracted facts and evaluates them in two comprehensive case studies. Given temporal aligned facts describing the same event at different point in time, retrieved from different sources, our proposed fusion strategies enable reliable views on the event. We test the end-to-end accuracy of our complete framework in two comprehensive case studies in the domain of disaster management covering real-world data.

Chapter 6 summarizes our findings and highlights future research directions.

1.3 Own Prior Work

The entire framework was sketched in [30], including the retrieval module (Chapter 2), our approach for estimating document publication dates (Section 3.1), and the fusion

module (Chapter 5). We reported evaluation results for the fusion stage only, covering solely the earthquake case study. All implementations, evaluations and the writing were performed by Döhling. Leser supervised the work and revised the manuscript.

Some approaches presented in extraction module (Chapter 4)—dictionaries, token neighborhood, and pattern matching in dependency graphs—were presented in [29]. We reported evaluation results for a subset of our earthquake corpora. All implementations, evaluations and the writing were performed by Döhling. Leser supervised the work and revised the manuscript.

Evaluation results for the domain robustness of extraction models (Section 4.6.4) were presented in [31]. All implementations, evaluations and the writing were performed by Döhling. Lewandowski contributed to the corpus annotation and revised the manuscript. Leser supervised the work and revised the manuscript.

2 Retrieving Event-relevant Documents

Finding material, such as documents, images, or videos, satisfying an information need from within large collections is the goal of information retrieval [81]. Using the web as source of information requires accessing the stored documents, as they contain the demanded information. But these documents are not on-hand, given the decentralized nature of the web. Instead, potentially useful documents satisfying information needs—the relevant documents—are located on different websites, numbering in the millions[1]. Accessing these sites and retrieving the relevant documents can basically be archived in two ways: directly by self-crawling or indirectly by search engines.

Self-crawling the web means automatically accessing all sites by following available hyperlinks from seed documents and downloading all documents found into a local document store. Having the documents on-hand, we can index their content and apply an established relevance model, such as the vector space model [115], to select the relevant ones for arbitrary information needs [81]. These information requests are usually formulated as textual queries, e.g. 'olympic games 2016 swimming' to retrieve documents about the swimming events at the 2016 Summer Olympics. Also, knowing the documents' fetch date can be beneficial to align the documents in time, further discussed in Chapter 3. Given the sheer size of the web, crawling in the general case is a laborious and costly task, especially if constantly repeated to capture new documents thus current information. To reduce the complexity, we can limit crawling to a manually curated list of sites potentially providing relevant documents [120]. Another possibility is to follow only hyperlinks which potentially lead to relevant documents, called focused crawling [19]. The decision which links to follow or not is based on link classifiers, trained on manually provided reference data. Both specializations lead to reduced document collections potentially supporting only a limited number of information needs.

Alternatively, we can utilize existing crawling infrastructures by invoking (commercial) search engines. They enable access to millions of documents from different sites via a single search interface [14]. Again, information request are usually formulated as textual queries, demanding appropriate search terms. Utilizing established search providers has the advantage of potential low latencies between the publication and the retrieval of new articles. They are able to adapt their crawling frequency to the number of updates per site, helping to analyze current events.

Processing events by means of our framework starts with retrieving event-relevant documents, containing the requested information. This first event-processing stage is fulfilled by the retrieval module, described in this chapter. It takes an event as input and returns a set of event-relevant HTML documents found on the web (see Figure 1.2 on page 4). These documents are then examined by the subsequent framework modules, described in the following chapters.

[1] http://www.internetlivestats.com/total-number-of-websites/

In this thesis, we classify documents as relevant if they report on the event, e.g. an earthquake, describe its impact, e.g. infrastructural damages, or cover closely related events, e.g. aftershocks related to an earthquake. We furthermore define an event as a triple [*type*, *location*, *date*]:

- The *type* refers to the event type, e.g. earthquake, flood, etc.
- The *location* refers to the area effected and can be a GPS coordinate or a list of toponyms, e.g. names of villages, cities, or countries. If *location* is given as GPS coordinates, our framework offers routines for generating corresponding toponyms by reverse geocoding employing GeoName's webservices[2].
- The *date* refers to the start date of the event in UTC.

For example, the Philippines earthquake in February 2012[3] can be described by

[earthquake, 9.964°N 123.246°E, 2012-02-06 03:49:16],

using geographic coordinates, and the river Danube floods in 2013[4] by

[flood, {Germany, Passau, Austria, Slowakia, Hungary}, 2013-06-01 10:00:00],

using toponyms.

We decided to rely on search engines providers to retrieve event-relevant documents, using adapters for Bing's[5] and Google's[6] search APIs. Self-crawling the web is not feasible for us and limiting crawling to specific sites potentially misses local media offering the most detailed information. Given a specific event, locale sites may provide extensive coverage, but these sites change with the event type and location. Curating a list of relevant sites would require manual work for each event, but we are interested in as little manual input as possible to process events by means of our framework.

Accessing the web via search engines demands appropriate, i.e. event-specific keyword queries. Moreover, as API providers, such as Google or Bing, limit the number of returned results per query, we require multiple adequate queries to broaden the number of relevant documents retrieved. These queries are created by the query generator (Section 2.1). It takes an event as input and combines event-specific terms to queries, derived from the event's attributes *type*, *location*, and *date*.

We cannot expect that all returned documents for the generated queries are event-relevant. Search providers may apply query reformulation [47] and/or sophisticated ranking to incorporate recency and diversity into the search result [33, 79], leading to potentially irrelevant search results. Returned documents may match only a subset of the applied search terms, reporting on the event type in general instead on specific events. Furthermore, synonyms and/or homonyms may interfere with the applied search terms. Therefore, we added document filters into our framework (Section 2.2), preventing potentially irrelevant documents containing misleading information from further processing.

[2] http://www.geonames.org/export/reverse-geocoding.html
[3] https://en.wikipedia.org/wiki/2012_Visayas_earthquake
[4] https://en.wikipedia.org/wiki/2013_European_floods#Danube_basin
[5] https://datamarket.azure.com/dataset/bing/search
[6] https://developers.google.com/custom-search/

Evaluating our retrieval approach in the general case is hardly possible, since we lack of gold standards, i.e. all relevant documents for a number of arbitrary events. Instead, we focus on the retrieval scenario set in the case studies covered in Chapter 5: gathering event-relevant documents months after a natural disaster. Section 2.3 reports recall and precision results for one example event, demonstrating the general feasibility of our retrieval approach. We discuss our findings in Section 2.4 and conclude with related work (Section 2.5).

2.1 Query Generator

The purpose of the query generator is to automatically create appropriate keyword queries based on a given event, defined by type, location, and date. Clearly, these queries should result in high document recall and precision. Recall is the fraction of found documents among all relevant ones that exist, precision the fraction of relevant documents among all found ones [81]. For retrieving event-data, recall also has a temporal dimension, as we are interested in retrieving documents covering the full period of the event, e.g. days or weeks, dependent on the type of the event.

Given the three event arguments, our approach for automatic query formulation is to combine *type* terms, *toponyms*, and *date* terms. For example, using the *type* term 'earthquake', the country name and the year of the 2012 Philippines earthquake creates the basic query 'earthquake Philippines 2012'. As search API providers, such as Google or Bing, limit the number of results returned, sending only one query is insufficient to retrieve the full event coverage from the web. In order to increase the recall, i.e. the number of relevant documents returned and the time period covered, we require additional semantic "similar" queries. These queries differ syntactically, but target the same event, e.g. 'quake Philippines February 2012' given the basic query 'earthquake Philippines 2012'. Our query generator uses methods to automatically create such queries by incorporating alternative and/or additional terms for all three arguments, a technique called query expansion [139].

- For *type*, search terms have to be provided by the user for each event type as part of the framework configuration. Finding terms describing the type of events is highly type-specific and should be conducted by inspecting type-relevant and irrelevant documents. Other potential sources for type-specific terms are ontologies, taxonomies, or thesauri [13].

- For *location*, we apply the country name, automatically derived from the GPS coordinates or the given list of toponyms, as basic search term. Additional toponyms for expanding the basic term are sourced from the given list of toponyms. If desired, further toponyms are automatically extracted from event-relevant documents by applying Stanford NER [44], a named entity tagger supporting toponyms. These seed documents may originate from initial search results using the basic query terms or may be provided by the user as part of the framework configuration.

- For *date*, we automatically derive the year as basic query term. Beside the year, we observed that the names of the month and the weekday of the beginning of

an event are often contained in relevant documents. We apply both names as automatically derived *date* expansions.

Each generated query combines one *type* term with the basic *location* and *date* term, optionally expanded by additional location and date terms. For instance, applying the *type* terms {'earthquake', 'quake'}, the *toponyms* {'Philippines', 'Visayas'} and the automatically derived *date* terms {'2012', 'February', 'Monday'} results in 12 queries for the Philippines earthquake:
'earthquake Philippines 2012', 'earthquake Philippines February 2012',
'earthquake Philippines Monday February 2012', 'earthquake Visayas 2012',
'earthquake Visayas February 2012', 'earthquake Visayas Monday February 2012',
'quake Philippines 2012', 'quake Philippines February 2012',
'quake Philippines Monday February 2012', 'quake Visayas 2012',
'quake Visayas February 2012' and 'quake Visayas Monday February 2012'.

2.2 Document Filters

As we cannot expect that all returned documents for the sent keyword queries are event-relevant, we complement our retrieval approach by configurable document filters. Search providers may apply query reformulation [47] and/or advanced ranking to incorporate recency and diversity into the result [33, 79], causing potentially irrelevant results. Applying our proposed document filters prevent potentially irrelevant documents containing misleading information from further processing. As a side effect, reducing the number of documents to process reduces the overall runtime of the framework as well. The document filters are independent of specific events, depending solely on document properties and the event type. These filters are:

Rank Filter Dropping documents originating from search results beyond a specified rank.

Blacklist Filter Dropping documents from specified hosts or domains. This filter is of great use to filter out sites with a potential low relevance for event coverages, e.g. `sourceforge.net`. Blacklists are manually curated.

Host-only Filter Dropping documents with an URL having an empty path or query part. So `//example.com` will be removed, `//example.com/?articleId=123` not. For example, when focusing on news-like articles, we suspect that URLs of relevant documents have such a non-empty path or query part.

Non-unique Filter Dropping documents that have been returned in searches for different events, presuming irrelevance to all events. For example, searching for specific earthquakes via queries containing the keyword 'earthquake' usually also returns results covering earthquakes in general, e.g. lists of earthquakes[7].

Size Filter Dropping documents whose HTML code exceeds a specified size limit.

[7] `https://en.wikipedia.org/wiki/List_of_earthquakes_in_2012`

Word Filter Dropping documents whose content is missing any term from a specified list of terms. This filter can be used to enforce the appearance of (parts of) the applied search terms in the returned results. The reasons for this filter are two general observations: (1) Search engines also return results not containing all query terms and (2) event-relevant documents contain at least one event type-specific key word. For example, we observed that the vast majority of earthquake reports contain the (sub)string 'quake'.

There are further document filters available in subsequent framework modules, as they depend on information unavailable at the retrieval stage. These filters are described in Section 3.4, 4.3, and 5.2.

2.3 Evaluation

Evaluating retrieval approaches in terms of recall and precision is a challenging task. Calculating recall values in general would require knowing all relevant documents for arbitrary events. Determining precision values usually requires manually checking all retrieved documents for relevance, if no external relevance criteria is available. Both, knowing all relevant documents or checking all retrievals, is hardly manageable given the sheer size of document repositories like the web.

Instead, we focus on the retrieval scenario set in the case studies presented in Chapter 5: gathering event-relevant documents months after a natural disaster. We evaluated our proposed retrieval approach for one example event, the before mentioned Philippines earthquake in Feb 2012. This setup aims at testing the general feasibility of our retrieval approach and the benefit of query expansion:

(1) Do we find relevant documents on the web based on automatically generated queries sent to search engines?

(2) Does sending multiple queries successfully increases recall values?

(3) Does filtering reduces the number of irrelevant documents while preserving relevant ones, i.e. increasing precision and keeping recall values?

We automatically collected multiple corpora by means of the retrieval module, with and without query expansion (Section 2.3.1). Recall estimations for these corpora are based on a manually collected set of relevant documents and precision estimates for some of the corpora are obtained by manual checks. For each corpus, we report evaluation results for both module stages: retrieval and filtering (Section 2.3.2). Document filtering here applies a set of filters simulating those utilized in the case studies presented in Chapter 5:

- the host-only filter,
- a size filter set at 500 kB, a limit determined by page sizes found in an external document collection, and
- a word filter enforcing the (sub)string 'quake' within document contents.

Table 2.1: Statistics for the retrieved corpora; *Documents* are based on unique URLs returned by the APIs, *Filtered* quote the number of documents after filtering. We also report the filtering effect on *REF*, although this filtered version is never applied in our evaluation. *REF* loses one document reporting on a tsunami scare without using the string 'quake' within the content.

	REF	BAS_{Bing}	BAS_{Google}	EXP_{Bing}	EXP_{Google}	WIK_{Bing}
Queries	—	1	1	11	11	78
Expanded?	—	No	No	Yes	Yes	Yes
Retrieved (2012)	Feb	Sep 9	Nov 5	Nov 5	Nov 5	Apr 19, 2013
Documents	177	99	99	455	528	1700
Filtered	(176)	96	97	431	482	866

Table 2.2: Distribution of the 177 documents in the reference corpus *REF* across sites.

Site	Documents
`inquirer.net`	61
`philstar.com`	31
`abs-cbnnews.com`	29
`heraldsun.com.au, news24.com, voanews.com`	6
`google.com`	5
`cnn.com, indianexpress.com`	4
`ph.news.yahoo.com, reuters.com`	3
`abc.net.au, bbc.co.uk, cbsnews.com, hindustantimes.com, ndtv.com, stuff.co.nz, timesofindia.indiatimes.com, usatoday.com`	2
`abcasiapacificnews.com, newsday.com, npr.org`	1

2.3.1 Data Sets

In the aftermath of the Philippines earthquake in Feb 2012, we created several corpora both manually and automatically by means of the retrieval module (Table 2.1). These corpora aim at investigating the general feasibility of our retrieval approach.

Reference Corpus REF Immediately after the earthquake, we manually monitored 22 sites reporting on the event and collected 177 relevant documents—i.e. URLs—over a period of three weeks. These 22 sites were selected based on search results derived manually shortly after the event and include global as well as local media (Table 2.2). We manually monitored these sites to capture new articles, leading to the overall 177 documents, forming our reference corpus *REF*. Surly, this corpus represents only a small sample of all relevant documents posted on the web covering the Feb 6th earthquake. Its purpose is to provide a basis to investigate the recall for subsequent sets of documents automatically created by the retrieval module.

Table 2.3: Automatically derived recall results for the generated corpora at the retrieval and filter stage (*Filtered*), based on the documents in *REF*.

	BAS_{Bing}	BAS_{Google}	EXP_{Bing}	EXP_{Google}	WIK_{Bing}
Recall_{REF}	3.4 %	1.1 %	9.0 %	9.6 %	8.5 %
Filtered	3.4 %	1.1 %	9.0 %	9.6 %	8.5 %

Basic Corpora BAS In autumn 2012, we queried both search APIs with the basic query 'earthquake Philippines 2012', generated automatically from the *type* term {'earthquake'}, the *toponym* {'Philippines'}, and the *date* term {'2012'}. The queries resulted in two corpora: BAS_{Bing} and BAS_{Google}. The purpose of these two basic corpora is to evaluate, if retrieving relevant documents by using keyword queries and search APIs works at all.

Expanded Corpora EXP Also in autumn 2012, we queried both search APIs with ten additional queries generated automatically by combining two *type* terms {'earthquake', 'quake'} with five *toponyms* {'Visayas', 'Negros Oriental', 'Cebu', 'La Libertad', 'Guihulngan'}, concatenated with the *date* term {'2012'}. These toponyms were manually derived from the documents in *REF*. Together with the basic query, these eleven queries resulted in the corpora EXP_{Bing} and EXP_{Google}. The purpose of these two expanded corpora is to evaluate, if applying multiple (expanded) queries in fact increases the recall.

Wikipedia Corpus WIK Furthermore, we queried the Bing API in April 2013 with 78 fully automatically generated queries, including derived *location* and *date* strings. These queries were based on the *type* terms {'earthquake', 'quake'}, the event's wikipedia page as seed document for automatically extracted *toponyms*, and automatically derived *date* terms, i.e. {'2012', February', 'Monday'}. This corpus is called WIK_{Bing} and is part of the multi-event data set generated in the case studies covered in Chapter 5. In other words, WIK_{Bing} represents an example outcome of the setup applied at the case studies. The purpose of this corpus is to evaluate the general feasibility of our retrieval module for these case studies.

Due to API limitations and presuming decreasing relevance for low-ranked documents, we retrieved only the top 100 results for each query. To apply the word filter after retrieval, we converted the document's HTML code into text strings, i.e. title, description, and content. The first two are defined by their respective HTML tag, whereas the actual page content is extracted by Boilerpipe [69].

2.3.2 Experiments & Results

Based on the *REF* corpus, we measured the recall without ($BAS_{Bing,Google}$) and with query expansion ($EXP_{Bing,Google}$, WIK_{Bing}) after both module stages: retrieval and filtering. Our results show that the recall increases considerably with the proposed expansion strategy, resulting in 2.3 % (avg) without compared to 9.3 % (avg) with expansion (Table 2.3). Comparing recall values for EXP_{Bing} (9.0 %) and WIK_{Bing} (8.5 %)

Table 2.4: Manually derived precision results for the basic corpora at the retrieval and filter stage (*Filtered*). Precision_{Event} refers to the event, i.e. the Feb 6$^{\text{th}}$ earthquake, precision_{Query} to the basic query 'earthquake Philippines 2012', i.e. all Philippines earthquakes in 2012.

	BAS_{Bing}	BAS_{Google}
Precision_{Event}	75.8 %	29.3 %
Filtered	78.1 %	29.9 %
Precision_{Query}	91.9 %	90.9 %
Filtered	92.7 %	92.8 %

indicate, that fully automatically generated queries perform equally to those with manually provided toponyms. Filtering reduces the number of documents substantially, e.g. −49 % for WIK_{Bing} (Table 2.1), while preserving the recall.

We manually determined precision results for the basic corpora ($BAS_{Bing,Google}$) at the retrieval and filter stage. We distinguish between an event-specific precision_{Event} and a query-specific precision_{Query}. Precision_{Event} refers to the event, i.e. solely documents about the Feb 6$^{\text{th}}$ earthquake count as relevant. As the basic query 'earthquake Philippines 2012' in fact covers multiple events, we alternatively count all documents about any Philippines earthquake in 2012 as relevant, leading to precision_{Query}. Precision_{Event} for the basic query ranges from 29 % to 76 % (Table 2.4). Precision_{Query} are approximately 91 %. Filtering slightly increases all precision values by +1.4 pp on average.

2.4 Discussion & Summary

We described the automatized retrieval of event-relevant documents from the web, the initial step of our framework. Given an event, defined by its type, location, and start date, we automatically generate event-specific keyword queries sent to search engines APIs. We utilize query expansion to overcome quantitative API limitations and apply filters to reduce the number of unnecessarily processed documents later in our framework. Sending multiple queries to established search providers and retrieving search results up to rank 100 potentially also help to mitigate the negative effects of "fake news" and "filter bubbles".

Our evaluation focused on the retrieval of relevant documents for past events as required by the information fusion experiment in Chapter 5. The results indicate that utilizing search engines via automatically generated queries effectively return event-relevant documents and that applying multiple queries leads to recall increases. Having a high recall is important in early stages of pipeline architectures, such as those supported by our framework. Missed relevant documents, thus their contained information, can never be recovered in later stages, whereas irrelevant documents might subsequently be filtered out. Although all pages cited in *REF* were still online during the retrieval experiments, we measured a relatively low recall of around 9 % in regard to *REF*. This is probably caused by a combination of the following aspects, interfering with automatic, recall-centric document retrieval:

Limited number of search results Our experiments limit search results to the Top 100 due to API limitations and presumed decreased relevance beyond this limit. As a consequence, matching documents ranked below are refused.

Unknown ranking Search providers mix their results as humans usually expect recency and diversity when searching for information covering events, especially among the top-ranked results. Therefore providers take different sources into account, avoiding too many hits from the same site within the top 100. This conflicts with the general pattern of local media reporting the most extensive on local events. For example, 61 of the 177 documents in *REF* are sourced from the Philippine media site `inquirer.net` (Table 2.2). It is reasonable that providers avoid returning all these 61 documents among the top 100. For BAS_{Bing}, the number of different sites returned is 77 with an average number of results per site of 1.3, led by `theextinctionprotocol.wordpress.com` with 3 hits, returning no results for `inquirer.net`. For BAS_{Google}, the number of different sites returned is 54 with an average number of results per site of 1.8, led by `youtube.com` with 11 hits, returning 2 results for `inquirer.net`. Search providers also potentially favor recent documents over past documents, as the latter might be outdated thus potentially less relevant to human information seekers. Furthermore they include results that match only a subset of the given query terms, e.g. returning documents about earthquakes in general.

Interference with similar events All retrieval experiments were conducted months after the targeted event. In the meantime, similar events occurred influencing the top 100. For example, on August 31st there was another earthquake striking the Philippines[8]. It reached an even higher magnitude than the evaluated Feb 6th earthquake, causing less damage. 6 out of the 99 documents in BAS_{Bing} and 54 out of 99 in BAS_{Google} cover this interfering event. This effect is also caused by the impreciseness of the applied basic query, further discussed below.

Valuing the measured precisions of 76 % (BAS_{Bing}) and 29 % (BAS_{Google}) in regard to Feb 6th earthquake requires similar considerations. As mentioned before, there was an interfering earthquake on Aug 31st, matching the investigated basic query 'earthquake Philippines 2012' too. Both *BAS* corpora contain results covering the second earthquake, lowering the evaluated Feb 6th-specific precision significantly. Later retrieval causes BAS_{Google} results to be more effected than BAS_{Bing} results. This is a side effect of our retrieval approach: The combination of potentially imprecise keyword queries and unknown ranking by search providers will hardly generate 100 % precision. Nevertheless, non-event-relevant documents do not harm the output of the framework in general, as we later apply a publication date filter (Section 3.4). In this specific case, its application raises both precisions to 100 %.

Considering both recall and precision, we assume that the relatively low recall does not generally imply an overall information cutoff within our framework. The measured high precision shows that other relevant documents ($\notin$ *REF*) are contained in the search results. Regarding the event processing in our framework, we are more interested in the contained information than the documents themselves. In other words, we should measure information recall instead of document recall, the former being even more

[8] `https://en.wikipedia.org/wiki/2012_Samar_earthquake`

difficult to obtain. Assuming redundancy across documents on the web in terms of contained information, we suspect that the presented retrieval approach is capable of retrieving a sufficient amount of event-relevant information.

2.5 Related Work

Retrieving relevant documents via search engines can be seen as a similar problem as accessing the so-called "hidden" or "deep" web [12]. The term refers to sites not directly browsable, for instance product data bases. In both cases, we have only indirect access to the underlying repository by using search forms, requiring adequate queries to gather the demanded documents. The purpose of hidden-web crawlers [105] is to gather as many "hidden" documents as possible. Depending on the capabilities of the present search form, transmitted queries may consist of simple keywords [10, 141] or domain specific predicates [35, 143], for example "price range" or "color". For keyword-based forms, popular terms—potentially returning many documents—are transmitted, selected from dictionaries or automatically derived from corpora or initial search results. In contrast to our work, these approaches are agnostic towards specific events or entities, focusing solely on maximizing the number of retrieved documents.

For searching event-relevant document on the "visible" or "surface" web, employing search engine APIs as in our work is typically limited by the number of results returned per query. Retrieving more relevant documents beyond this limit can only be achieved by additional "similar" queries or self-crawling. Given initial queries, Thelwall proposed query splitting as one solution for generating new queries based on the same keywords [125]. The initial queries are modified by adding or subtracting another term; subtracting here refers to "should not contain" and is often expressed by a preceding '–'. Augmented terms might be generic, e.g. found in external corpora, or derived from initial search results. An alternative technique is query expansion [18], originally designed to improve the recall by adding potentially relevant terms to basic queries. Similar to query splitting, expansion has the (side) effect of altering the rankings, potentially returning previously unseen results below API limits, utilized in our retrieval approach.

Complementary to query reformulation is the task of generating initial queries describing particular entities or events. These queries should be derived from individual properties, demanding a generative process taking entities/events as input and returning adequate keywords as output. For instance, Endrullis et al. proposed query generators for product search in electronic marketplaces, utilizing shared (e.g. manufacturer) and unique properties (e.g. ISBN) [36]. This is similar to our approach, using the event properties *type*, *location*, and *date*. In order to retrieve related information to aid concept mapping, Leake et al. utilized map-specific terms to generate assisting search engine queries [74]. More recently, retrieving microblogs (e.g. tweets) led to the revival of the problem of transmitting event-specific keyword queries [95, 113, 130]. Utilizing author-generated tags—so called hashtags—is of great help here [132, 140]. These ad-hoc labels created by users, e.g. '#eqjp' for the 2011 Japan earthquake[9], identify event-relevant messages precisely, demanding automatic recognition of these labels in the message stream.

[9] https://en.wikipedia.org/wiki/2011_T%C5%8Dhoku_earthquake_and_tsunami

3 Aligning Documents in Time

Aligning documents in time is a crucial prerequisite for time-aware information retrieval and extraction, especially for highly dynamic document collections, such as the web [5, 17]. For instance, document timestamps may be applied to improve retrieval for temporal queries, such as 'fifa world cup 1990s' or 'earthquake japan 2011' [11, 51]. Temporal alignment on the web might be archived by evaluating announced publication dates, specifying last document updates/modifications. The HTTP's Last-Modified header[1] enables authors to announce these publication dates. In practice, this header is often unavailable as it requires a log of the server response during downloading. Even if available, it might be unreliable as servers frequently return the current date [96], probably due to search engine optimization. The fetch date—given its availability—does not help either, as it may be arbitrarily incorrect, depending on the time between the publication and the page download. However, if crawling at a high frequency is feasible, fetch dates may provide fair approximations for new documents. In case of (X)HTML documents, meta elements[2] allows specifying the publication date [97], but this suffers from similar limitations as the HTTP header. Given the document's URL, rough estimates for publication dates might be achievable by searching web archives [61] or evaluating the vicinity, i.e. timestamps of incoming or outgoing links [96, 114]. Given a language model for the targeted time span and language, the document's words and their frequencies might permit determining the year or the day of week [27]. These approaches suffer from depending on external data with unknown availability and granularity.

A reliable and broadly available information source for publication dates of web pages is the HTML body itself. It often contains human-readable strings, such as 'Posted February 6th, 2012 at 4:35 am', especially when describing events in time. Utilizing these strings for temporal alignment of documents requires their recognition and parsing, facing multiple candidates as well as diverse date formats and page structures. Figures 3.1 and 3.2 show two web pages as examples. They allow identifying at least five disjoint classes of dates that appear at various places on pages and potentially interfere with recognizing the targeted publication date:

(1) The fetch date denotes the current date the moment the document was downloaded. It may differ arbitrarily from the publication date, depending on the time between the page's publication and the download. However, we expect the fetch date to be younger or equal to the publication date.

(2) Content dates refer to dates or temporal expressions mentioned inside the text blocks of the page, i.e. titles, captions, or paragraphs. This class contains absolute dates, such as 'November 12, 2014' (Figure 3.2), as well as relative expressions,

[1] http://tools.ietf.org/html/rfc2616#section-14.29
[2] http://www.w3.org/TR/html401/struct/global.html#h-7.4.4

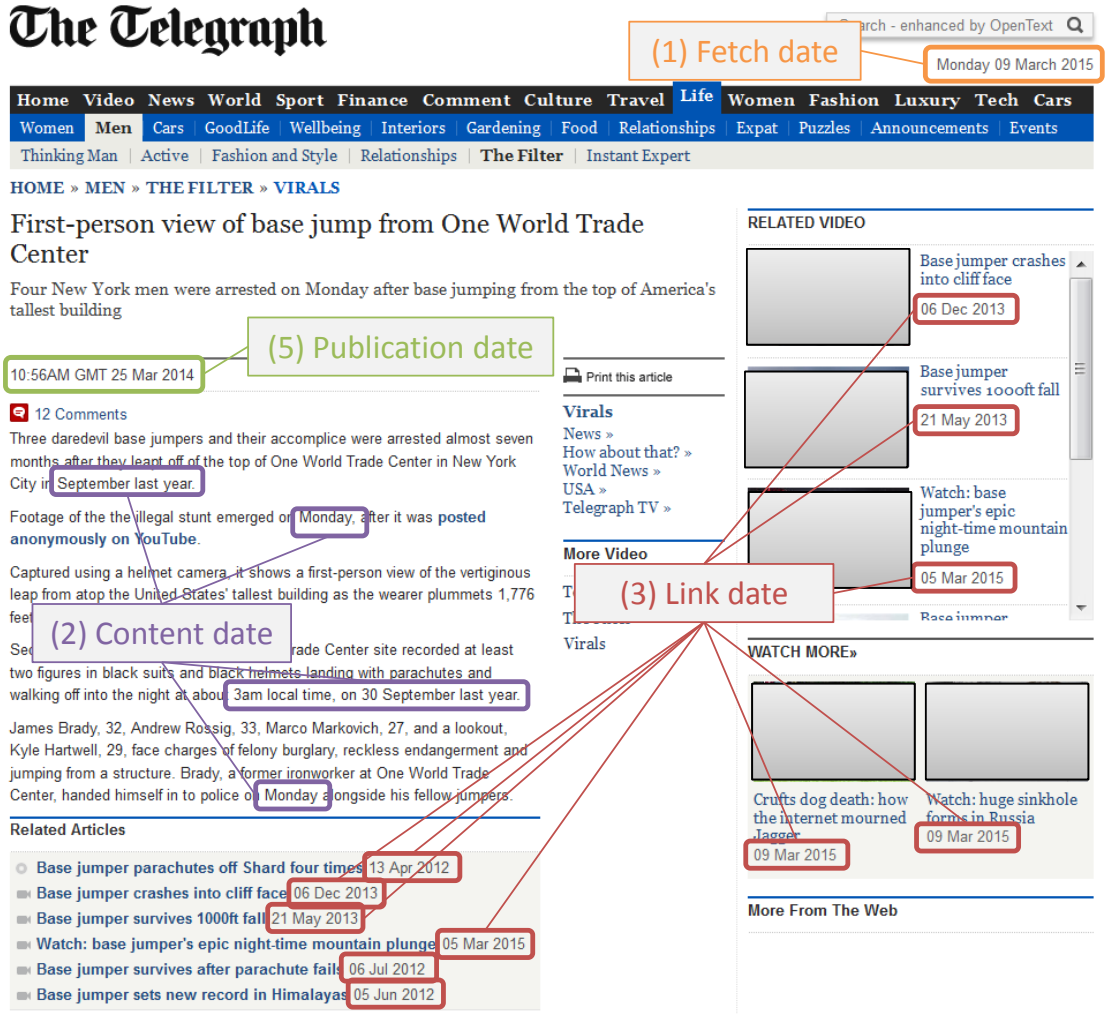


Figure 3.1: An example web page* presenting further—potentially interfering—dates beside the targeted publication date, using diverse formats.

*http://www.telegraph.co.uk/10720820/article.html

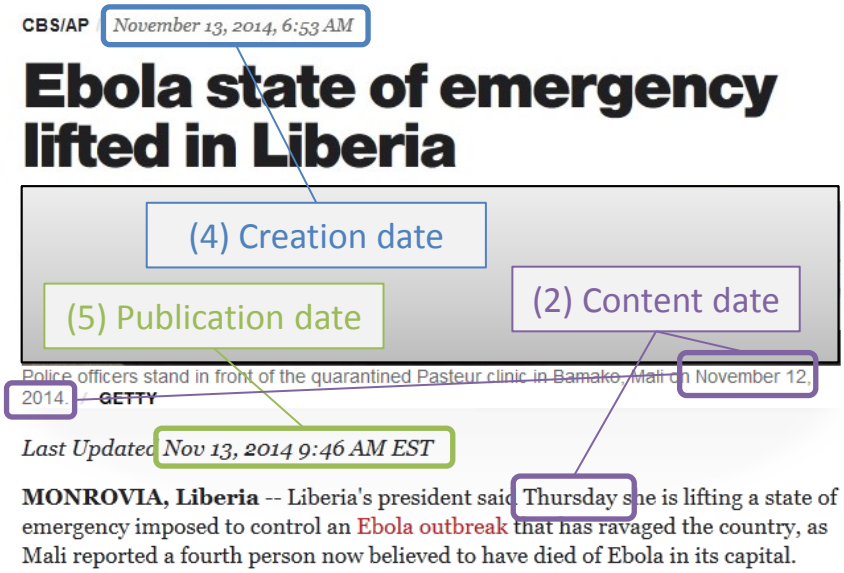


Figure 3.2: An example web page* presenting further—potentially interfering—dates beside the targeted publication date, using diverse formats.

*http://www.cbsnews.com/news/ebola-vaccine-trials-to-begin-in-west-africa-amid-threat-of-mali-outbreak/

such as '3am local time, on 30 September last year' (Figure 3.1). In relation to the publication date, content dates may mark arbitrarily dates in the past or the future. Temporal taggers have been designed to extract—even relative—temporal expressions from page contents [22, 42, 121]. However, we observed insufficient recall and accuracy at recognizing the publication date, tested on a sample set of web pages. Moreover, these taggers require the publication date as input to solve relative expressions, not providing it as dedicated output.

(3) Link dates refer to dates connected to linked pages, i.e. part of the link or the link caption. In relation to the publication date, link dates may mark arbitrarily dates in the past or the future (Figure 3.1). However, we expect link dates to be younger or equal to the fetch date.

(4) The creation date denotes the first page revision ("created at"). We expect the creation date to be older or equal to the publication date and younger or equal to the fetch date.

(5) The publication date denotes the last page revision ("updated at"). We expect the publication date to be younger or equal to the creation date and younger or equal to the fetch date.

For timestamping documents, we are interested in recognizing the publication date. On the web, the publication and creation date for documents, e.g. news articles, might

differ, as news providers tend to update the content without changing the URL. Consequently, the publication date refers to the document's content, whereas the creation date refers to (the first appearances of) the document's URL. We use both terms synonymously if only one of the two dates is present.

After recognizing the string denoting the publication date, we need to parse it. Parsing date expressions implies various challenges, especially ambiguous date formats and time zone recognition. In the UK, for instance, the expression '06/02/2012' denotes 6$^{\text{th}}$ February, whereas in the US it is interpreted as June 2$^{\text{nd}}$. This variance is later referred to as "UK/US ambiguity". Determining the correct publication date also requires handling time zone expressions. While terms like '+0530' or 'GMT' uniquely identify time zone offsets, there are many abbreviations which do not. For instance 'PST' denotes both, Pacific Standard Time (UTC−08) and Philippine Standard Time (UTC+08), differing by 16 h. Depending on the demanded accuracy and granularity, such ambiguities might have a significant impact on the results of subsequent tasks. Identifying solely the correct day might be sufficient for arranging documents in chronological order of days [58]. However, it is insufficient for reports on (crisis) events with frequent changes in available information, requesting precise alignment to the minute (Chapter 5).

In this chapter, we describe methods for the temporal alignment of (web) documents, i.e. time-stamping documents with their publication date. These methods are part of the temporal alignment module in our framework. The module takes a set of documents as input, determines their publication date, and forwards the time-stamped documents to the subsequent modules (see Figure 1.2 on page 4). We compare three alternative approaches for aligning documents in time: PcDE, CarbonDate, and DCTFinder. Each approach estimates document timestamps based on different information and/or a different method. Our rule-based approach called PcDE applies textual patterns, such as 'yyyy-MM-dd HH:mm', to identify publication/creations dates shown in web pages (Section 3.1). In contrast, CarbonDate ignores the document itself and uses incoming links to estimate creation dates (Section 3.2). DCTFinder is also content-based, but applies machine-learning techniques to find contained creation dates (Section 3.3). Evaluation results for the three on an elaborate corpus of English web pages are given in Section 3.5, allowing analyzing their strengths and weaknesses. We discuss our findings in Section 3.6 and conclude with related work (Section 3.7).

3.1 PcDE: Rule-based Date Estimator

We created a rule-based approach called PcDE[3] for temporal alignment of web pages. PcDE uses solely the pages' source code to determine their publication or creation date. It uniquely combines three aspects of temporal alignment: providing granularity to the second (if seconds are provided by the page), focusing on last page revisions, and being independent from external information. PcDE also addresses date format disambiguation and time zone recognition. We divide the identification of exact publication dates into two steps: recognizing all timestamps, forming the set of candidates (Section 3.1.1), followed by deciding which candidate is the publication date (Section 3.1.2).

[3] *P*ublication and *c*reation *D*ate *E*stimator

3.1.1 Candidate Extraction

We first convert the HTML source of the page into text blocks—signaled by block-level elements (`h1`, `p`, `div`, etc.)—by applying Boilerpipe [69]. This library additionally discarding non-content elements, such as advertisements or comments. Next, date candidates are extracted by matching date patterns against all text blocks having 150 characters at maximum. This limit is intended to prevent content dates from becoming candidates (see date class (2) in the introduction of this chapter). Although we suspect a limited number of possible date formats in a given page, specifying a pattern for each possibility would be a cumbersome and error-prone task. Instead we apply date expression-specific stemming as a preprocessing step before the actual matching. Linguistic stemming refers to the process of reducing morphological variance of words, e.g. mapping 'example' and 'examples' to 'exampl' [70]. This reduction potentially eases matching processes afterward, e.g. requiring only the term 'exampl' to match 'example' and 'examples'. We adopt this strategy by removing all characters/words which do not belong to possible date expressions, reducing the number of required patterns substantially. For example, stemming

'February 6, 2012 -- Updated at 23:15 GMT'

as well as

'(UKN/ANN - New York) Published at: February 6th, 2012 | 23:15 (GMT)'

results in

'February 6 2012 23:15 GMT',

matching the pattern[4]

'MMM d yyyy HH:mm z'.

Our rule-based stemmer keeps month name, time zone expressions, and numbers having two or four digits, possibly forming years, days, minutes, etc. It distinguishes between characters forming valid field separators, such as ':', separating for instance hours and minutes, and other punctuations, such as '|', getting deleted. Over-stemming refers to removing to many characters, leading to erroneously mapping of unrelated words/instances onto each other, e.g. 'policy' and 'police' onto 'polic' [103]. For our date stemmer, this refers to removing words between two partial and unconnected date expressions, than forming a valid but false date expression. For instance, stemming 'The new bridge will be opened in 2023. The old bridge will be demolished on Feb 6th at 12.00 pm.' would result in the false stem '2023 Feb 6 12.00 pm', matching the date pattern 'yyyy MMM d h:mm aa'. To avoid over-stemming, we keep word sequences of length ≥ 6 whereas shorter sequences, such as 'Updated at', are deleted.

After stemming, we match 1052 patterns, such as 'yyyy-MM-dd HH:mm:ss', to extract and parse all date candidates. These patterns were built by combining 21 day patterns and 10 time patterns, accompanied by a time zone pattern. Each day pattern is combined with each time pattern in both possible orders, optionally followed by a time zone notation: "day time" or "time day" as well as "day time zone", "time day

[4]Literals used refer to Java's SimpleDateFormat syntax `https://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html`

zone", or "time zone day". The day and time patterns—listed in Appendix A—were manually derived from 177 English articles from 23 hosts published in February 2012 (Table 3.1). These articles equal to the *REF* corpus in Chapter 2 and are later referred to as PcDE's development corpus. The parsing process works from left to right on all stemmed text blocks, trying to match each pattern at each text position. The precedence of patters is manually set, preferring long patters over short patterns. After a successful match, we skip all remaining patterns and restart the matching process at the subsequent text position.

Parsing ambiguous date expressions on (English) web pages, e.g. '06/02/2012', requires distinguishing between UK (dd/MM/yyyy) and US (MM/dd/yyyy) day format. To determine the appropriate format, we scan all text blocks for unambiguous expressions, such as '16/02/2012'. If none can be found, we inspect the top-level domain of the page's URL, often available as metadata. e.g. provided by the preceding retrieval process (Chapter 2). Pages from `com`, `net`, `org`, `tv`, or `us` trigger the US format, all others the UK format. If the URL is unknown as well, the US format is preferred for parsing. The determined format influences the precedence of applied patterns, i.e. favoring US patterns over UK patterns if we detect the US format and vice versa. In any case, the other patterns are used as subordinate patterns, applied if parsing with the preferred patterns fails, e.g. due to a misleading hint from the URL.

We also detect the time zone if present in the date expression. Numerical offsets ('+0530') are interpreted directly whereas textual notations ('PST') are recognized based on a list of abbreviations retrieved from Wikipedia[5]. To minimize the effect of ambiguous zone names, we apply their average offset based on the values listed on Wikipedia. If we cannot identify any time zone, we apply an optional, user-specified default zone—for instance derived from the event's location—or UTC.

3.1.2 Candidate Selection

By default, we select the first—according to the western reading direction—candidate found as the publication date of a web page. We exclude dates before 1995, acting as a pre-WWW boundary, a filter which should be removed if also historical texts are of interest. Future dates and dates after the fetch date—if available—are excluded as well. Additionally, we ignore all dates originating from text blocks having more than two candidates. If we detect two candidates in one text block or two candidates in total located in consecutive blocks, we suspect a combination of creation and publication date. In these cases, we propagate the more current one as publication date. In fact, this is the only rule in our approach that separates publication from creation dates. As a consequence, inverting it, i.e. preferring the older date, transforms our approach into an estimator for creation dates.

[5] `https://en.wikipedia.org/w/index.php?title=List_of_time_zone_abbreviations&oldid=516840413`

3.2 CarbonDate

The second method for temporal alignment we evaluate is CarbonDate[6], a web service based on the work of SalahEldeen et al. [114]. They propose utilizing a combination of the page's Last-Modified header and timestamped incoming links for temporal alignment. These so-called backlinks are hosted on third-party sites, such as URL shorteners. Hence, only pages already linked on these sites can be aligned, forming a crucial limitation not present in PcDE. Using the first appearance of URLs on these sites implies coarse granularity and restricts their approach to creation dates, ignoring updated contents. Moreover, their approach requires the document's URL for alignment, which might not be available.

3.3 DCTFinder

The third method we evaluate is DCTFinder[7], a Java library based on the work of Xavier [124]. Analogous to our approach, DCTFinder extract dates directly from page contents, but similar to CarbonDate focusing on creation dates. Furthermore, DCTFinder extracts only the day part of given dates, ignoring all time and time zone information. Comparable to PcDE, DCTFinder follows a two-step approach to determine creation dates: recognizing candidates and selecting the correct one. Candidates are found by labeling the tokenized page content with a Conditional Random Field (CRF) [72], utilizing lexical and structural features. Examples for language-dependent lexical features are month names, triggers ('puplished'), or anti-triggers ('comments'). Structural features capture information, such as the distance from the title. Only tokens neighboring numerical values are considered for labeling, resulting in text filtering analog to our stemming. After labeling, candidates are parsed using a set of pre-defined date formats. Among all candidates, the oldest one is returned as the creation date, optionally filtered by the fetch date. The UK/US format disambiguation is based on a explicitly specified format or the URL's top-level domain, defaulting to the US format. CRF models for English and French as well as their respective training data are available from the project's homepage[7]. We evaluate the English model, trained on 563 web pages from 376 hosts posted in the first half of 2008 (see Section 3.5.1).

3.4 Document Filters

We also included a configurable, publication date-based document filter in our framework. Setting it prevents documents created before or after the specified date from further processing. Given an event, such as the 2016 Summer Olympics, it allows to separate reports before the event from those during or after the event. Note that setting an (explicit) date filter implicitly revokes all documents without any recognized publication date as well.

[6] `http://cd.cs.odu.edu/`
[7] *D*ocument *C*reation *T*ime *Finder* `http://sourceforge.net/projects/dctfinder/`

3.5 Evaluation

We evaluated all three approaches for temporal alignment on a specifically designed corpus of English web pages (Section 3.5.1). The corpus is based on randomly sampled pages manually enriched by rare and potentially hard cases. Accuracies as defined in Section 3.5.2 are reported in Section 3.5.3, including measurements for CarbonDate. Results at day granularity, also testing DCTFinder, are presented in Section 3.5.4. Calls to the date estimators were parametrized with the page content and/or the URL, the fetch date, and UTC as default time zone. All experiments were conducted in January 2015.

3.5.1 Data Sets

Creating a corpus for assessing publication date extraction is a laborious task, particularly if aiming at covering as many hard—and probably rare—cases as possible. Selected pages should be as heterogeneous as possible, but rather in terms of date formats, page structures, etc. than topics or contents. Consequently, just crawling a couple of sites is inappropriate, as these pages are potentially homogeneous with dates uniformly created by a content management system. Instead, the corpus should include pages from various hosts. Using multiple pages from the same host is acceptable here, if their date format or page structure significantly differs. This is often the case for pages originating from different years. In addition, sites using ambiguous date formats demand two samples, pages with and without an actual ambiguous date expression. For instance, UK/US ambiguity exists for 36 % of all days within a year, legitimating to include both type of pages. Another aspect of the corpus-based evaluation is that hosts included in the training/development set[8] are ineligible for the evaluation. For example, using training pages from publishers like BBC or CNN—which is reasonable—should exclude them from the evaluation. From an application point of view this is suboptimal, as we are also interested in the expectable accuracy for hosts contained in the training set. We therefore report evaluation results for both, hosts contained ("known") and not-contained ("unknown") in the training/development set.

Our evaluation corpus consists of 116 pages originating from 91 hosts, designed to compare different approaches for temporal alignment among each other (Table 3.1). The corpus contains 68 randomly selected pages published in March 2013, collected by DCTFinder's author. It is complemented by 48 manually selected pages posted between 2009 and 2014, targeting the outlined challenges of publication date extraction. We manually annotated all publication and/or creation dates, constituting our gold standard. Identifying the correct dates is based on the HTML source, utilizing the human-readable date strings as well as available header metadata. Determining the correct time zones often requires inspecting the site a page is sourced from. 16 % of the hosts contained in the evaluation corpus also appear in PcDE's development corpus ("known hosts"), but having distinct URLs. For DCTFinder, the host-level intersection between its training corpus and our evaluation corpus is 26 %, also having distinct URLs.

[8] Both terms refer to non-evaluational parts of corpora. We use the term "training" for machine-learning-base methods, such as DCTFinder, that automatically train a model and the term "development" for methods based on the manual acquisition of rules/patters, such as PcDE.

Table 3.1: Corpus statistics

	PcDE development	DCTFinder training	Evaluation
Pages	177	563	116
Hosts	23	376	91
Published	Feb 2012	Jan–Jun 2008	2009–2014

All pages show a human-readable, absolute publication or creation date including a time expression. These dates are located inside the HTML content including `noscript` tags, but excluding tag attributes or `javascript` elements. 22 % of the pages contain both dates, with half of them showing only the publication date and providing the creation date as metadata. Although hidden from human readers, these creation dates are crucial for proper evaluations and are therefore considered for the gold standard as well. For pages containing both dates, these dates differ in 4.5 h on average with a maximum of 20 h, emphasizing the differentiation between the publication and creation date. 21 % of the pages utilize an ambiguous US/UK date format and of these, 63 % have an actual ambiguous date expression. 75 % of the pages report dates using a time zone offset deviating from UTC, often missing the time zone expression. Explicit time zones are shown in 47 % of the pages, with 15 % of them being ambiguous.

3.5.2 Evaluation Measure

We measure the quality of temporal alignments by assessing the difference between the expected timestamp t_{exp} set by the gold standard and the estimated timestamp t_{act} returned by the evaluated method. Awarding only returned dates that exactly match the gold standard, i.e. testing if $t_{exp} = t_{act}$, is inappropriate here. It penalizes non-extraction-based approaches, such as CarbonDate, that usually return dates not present in the page, unequal to the gold standard in most cases. Instead, we are more interested in how close the returned dates match the gold standard. We therefore extend the concept of true positives—the number of correctly identified positive instances—by adding a tolerance Δt to the matching decision, i.e. testing if $|t_{exp} - t_{act}| \leq \Delta t$. Given the tolerance Δt, we define accuracy at Δt as the portion of all pages (#pages denoting their number) whose timestamp difference is less or equal to the tolerance:

$$accuracy(\Delta t) = \frac{\#pages\ having\ \ |t_{exp} - t_{act}| \leq \Delta t}{\#pages}$$

This definition complies with the standard accuracy definition—portion of correctly identified among all instances—if applying the extended concept of true positives [82]. Beside allowing to evaluate non-extraction-based approaches, it also compensates false but nearly correct extraction-based results, e.g. returning the (close) creation date instead of the publication date.

3.5.3 Results

Publication Date

Applying PcDE to the evaluation corpus, we are able to report 31 % of the publication dates exactly to the second and 47 % to the minute (Figure 3.3a). This difference is caused by the common practice of omitting the seconds in the human-readable date string but providing it in the gold standard metadata. The fraction of correctly reported publication dates rises to 91 % if permitting 12 h tolerance. The reported dates differ for 2.6 % of the pages by more than 48 h and 3.4 % return no date. We also measured the recall at the first stage of our approach, i.e. the candidate extraction (Section 3.1.1). This number shows how often the targeted date string is among the extracted candidates, ignoring any subsequent interpretation or selection errors. For our evaluation corpus, this recall is 96 %, indicating a high coverage of the applied date patterns (Appendix A).

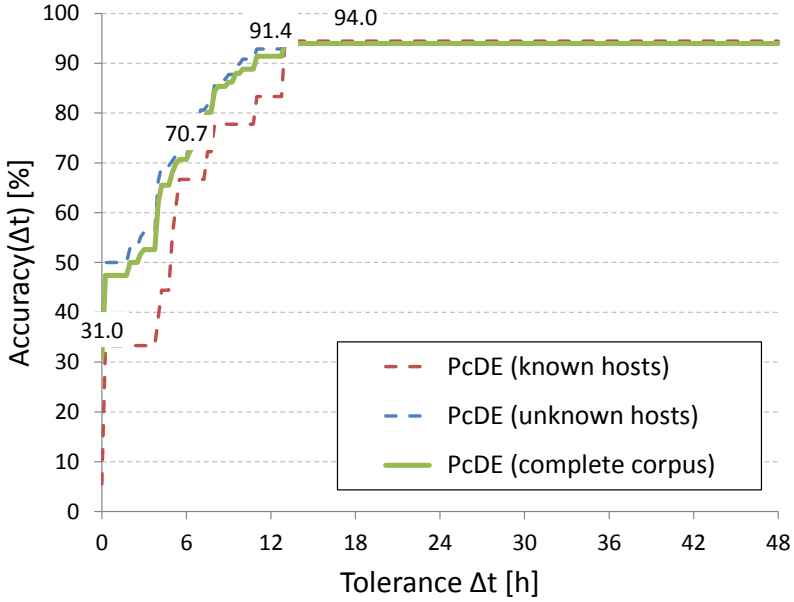
Creation Date

We further investigated the suitability of our approach for exact creation date estimation, additionally comparing it with CarbonDate (see Section 3.2). Regarding PcDE, returning creation dates instead of publication dates requires inverting the switch for two consecutive dates (see Section 3.1.2). We call the resulting creation date-oriented version PcDE_C. We assume midday for those dates returned by CarbonDate missing any time information. Applying PcDE_C for creation dates yields comparable accuracy to estimating publication dates with PcDE, deviating by approximately −1 pp (Figure 3.3b). PcDE_C outperforms CarbonDate by +20 pp accuracy on average on the entire interval of 48 h. Detailed results for CarbonDate are: 0.0 % accuracy to the second, 2.6 % to the minute, and 66 % at 12 h tolerance. For 8.6 % of the pages, dates differ by more than 48 h and 16 % return no date.

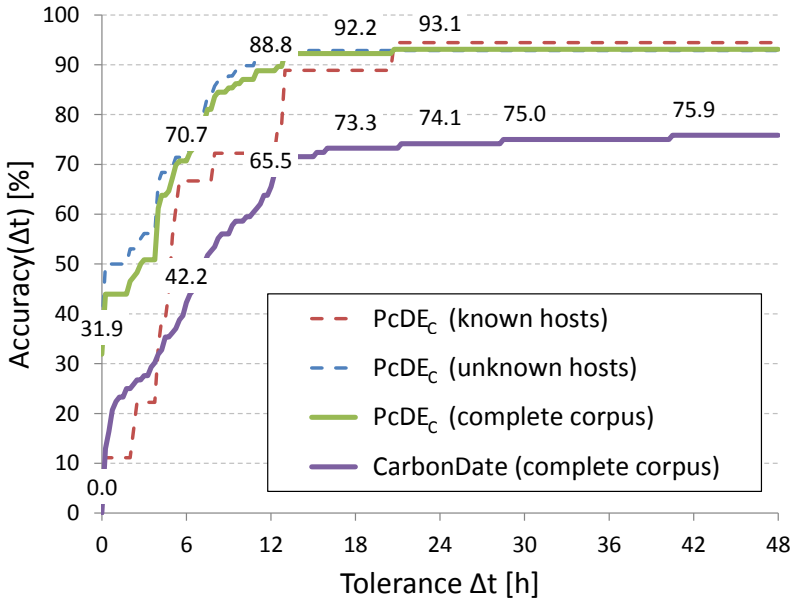
3.5.4 Day-only Results

Creation Date

In addition to exact temporal alignment, we also evaluated our approach at day-only granularity on creation dates. This allows us to compare it with DCTFinder (see Section 3.3). The evaluation requires truncating the time part of the dates in our gold standard, still respecting the time zone information. For example, the expression 'January 10, 2012 10:39 PM PDT' is treated as January 11^{th}, as this is the day part of the normalized date in UTC. Likewise, we truncate the time part of the creation dates returned by PcDE_C, calling this version PcDE_CD. Under zero days tolerance, PcDE_CD correctly identifies 83 % of the creation dates, whereas DCTFinder archives only 70 % (Figure 3.4). Both profit from increasing tolerances, yielding 93 % (PcDE_CD) and 87 % (DCTFinder) at one day tolerance. For 10 % of the pages, dates returned by DCTFinder differ by more than 48 h and 5.2 % return no date (PcDE_CD: 3.4 % and 3.4 %).



(a) Publication date evaluation



(b) Creation date evaluation

Figure 3.3: Results for the exact temporal alignment of the evaluation corpus, distinguishing between hosts appearing in the development corpus ("known hosts") and unknown host. Tolerances are calculated to the second and listed as hours for legibility. Plotted numbers are related to the entire evaluation corpus.

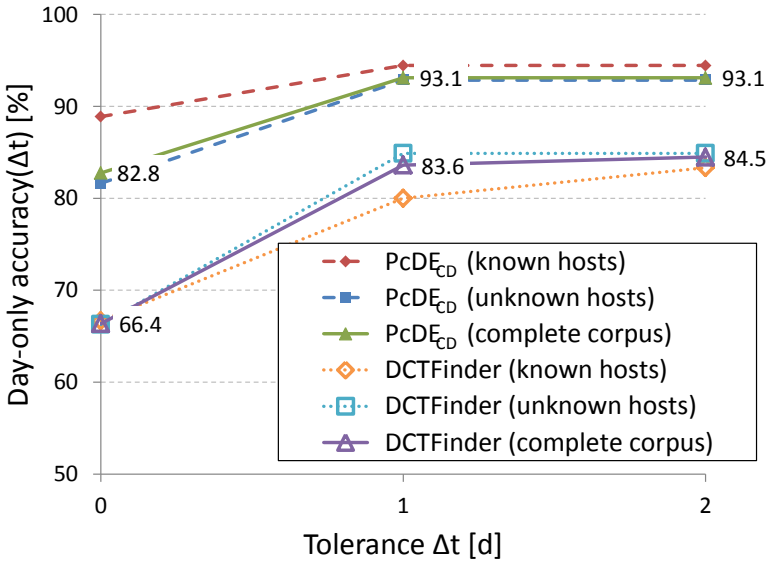


Figure 3.4: Results for the day-only alignment of the evaluation corpus, distinguishing between hosts appearing in the development/training corpus ("known hosts") and unknown host. Plotted numbers are related to the entire evaluation corpus.

3.6 Discussion & Summary

We compared three approaches for aligning web documents in time and evaluated them on a dedicated corpus of English web pages. Knowing the timestamp of documents is crucial for time-aware tasks downstream, such as information fusion (Chapter 5). Two of the approaches utilize third-party services/tools whereas the third—PcDE—represents our own work. PcDE uniquely combines three aspects of temporal alignment: providing granularity to the second, focusing on publication dates, and solely utilizing page contents. Despite its simplicity, it achieved accuracies of up to 91 % on English web pages. The key to PcDE's success is to limit the candidate extraction to those date expression that include time information (Figure 3.1). Still, 31 % of the evaluated pages generate at least two candidates (Figure 3.2), potentially interfering at the subsequent candidate selection. Evaluation results separated by "known" and "unknown" hosts indicated no bias towards previously seen hosts (Figure 3.3), negating information leakages between development and evaluation. In addition, PcDE$_{C}$ surpassed CarbonDate by +20 pp accuracy for the comparable task of estimating the exact creation date. The observed performance gap is mainly caused by unavailable external information, emphasizing the benefits of content-based approaches. We also tested PcDE$_{CD}$ on the task of resolving the days of page creation, yielding substantially better results than DCTFinder. However, CarbonDate and DCTFinder carry the advantage of being able to determine creation dates for pages reporting only the day. PcDE fails for these pages due to requiring a time expression. In addition, CarbonDate may provide estimates for documents missing any date information or even images, as it only utilizes their URL. Consequently, CarbonDate can only be applied for documents with known URL.

Table 3.2: Comparison of the characteristics of the presented approaches for temporal alignment

	CarbonDate	DCTFinder	PcDE
Offered granularity (up to)	second	day	second
Returns creation dates	yes	yes	yes
Returns publication dates	no	no	yes
Requires the document's URL	yes	no	no
Requires the document's content	no	yes	yes
Depends on external information	yes	no	no
Requires contained date expression	no	yes	yes
Requires contained time expression	no	no	yes
Respects time zones	yes	no	yes
Supports non-textual documents	yes	no	no
Is language independent	yes	no*	no**

* Trained models for English and French are available

** Numer. patterns might also work in other Latin character-based languages

All three evaluated approaches have their strengths and weaknesses (Table 3.2). Selecting the appropriate approach depends on the task and the targeted documents. For instance, day granularity should be sufficient for arranging documents in chronological order [58] or temporal scoping of facts [123, 133]. In contrast, reports on events with frequent changes in available information, such as natural disasters, require precise alignment to the minute or hour (Chapter 5).

3.7 Related Work

Aligning documents in time, also called temporal text classification [17], has been addressed before. Jatowt et al. tried to reconstruct the creation time of web content by comparing snapshots taken from web archives [61]. Consequently, their approach is limited by the coverage and recency of such archives, offering only rough timestamp estimates, if at all. Nunes et al. used the HTTP headers of linked pages—mostly outgoing—and embedded resources—for instance images—to determine the page's Last-Modified value [96]. Exploring the vicinity of web documents increased the availability of Last-Modified values from 53 to 86 %. They measured a correlation of 0.73 between the Last-Modified values of pages and their neighborhood, but no accuracies for estimated values were reported. SalahEldeen et al. utilized a combination of the page's Last-Modified header and timestamped incoming links ("backlinks"), hosted on third-party sites [114]. Using the first appearance of URLs on these sites limits their approach, called CarbonDate, to creation dates, ignoring updated contents. They were able to estimate creation dates for 76 % of 1200 sampled pages, including 33 % having the cor-

rect day. Results for CarbonDate on our data set are presented in Section 3.5.3. All of the above approaches suffer from similar limitations due their dependence on external information: unknown availability and coarse granularity. Moreover, they require the document's URL to gather these information.

Relying solely on page contents is one solution to overcome these limitations. Here we can identify two strategies: (1) applying language models and (2) extracting the publication date directly. For instance, Dalli et al. used temporal profiles of word frequencies to align documents in time [27]. The results of their evaluation is not directly comparable to our work as they determine the day of the week, the week, the month, etc. independently. For example, they correctly identified 32 % of the days of the week and 88 % of the years for English pages. A major drawback of using language models is the requirement of timestamped training data for the questioned time interval and the targeted language. Inoue et al. examined extracting the creation date directly from the page content, limited to the day part [58]. They start by scoring each contained date expression—identified by patterns—by a complex set of positive or negative factors. These factors are manually derived rules utilizing lexical and structural features. The date with the highest score is returned as creation date. Interpreting ambiguous date expressions correctly is achieved by calculating the differences between all contained dates for all interpretations. The interpretation with the lowest average difference is selected for the respective page, defaulting to the US format for single-date pages. They were able to correctly extract 91.3 % of the creation dates from English web documents. Strongly related to their work is the approach of Xavier, called DCTFinder [124]. It also tries to detect the day of page creation by inspecting all date expressions utilizing comparable lexical and structural features. In contrast to Inoue et al., these features are the input for token-based labeling per Conditional Random Field (CRF) [72]. Among all dates positively labeled, the oldest one is returned as the creation date, optionally filtered by the fetch date. The UK/US format disambiguation is based on a explicitly specified format or the URL's top-level domain, defaulting to the US format. It yielded an accuracy of 90.0 % on English web pages collected in March 2013. Results for DCTFinder on our data set are listed in Section 3.5.4. Although both approaches utilize the page content, they are limited to day-only granularity and creation dates.

4 Extracting Facts from Documents

Today, great amount of information are available in textual (and electronic) form. This includes newspapers or blogs, repositories, such as PupMed, encyclopedias, such as Wikipedia, or user-generated content on Twitter. Exploring the available information for any event or topic manually is no longer feasible. For example, querying search engines with 'olympic summer games 2016' results in more than 10 million potentially relevant web documents. It is impossible to read all of them, calling for automatic approaches. Information extraction (IE) as an area of research offers such automatic methods [116]. It studies the problem of extracting specific information from unstructured, natural language texts. IE has a long tradition in computer science, spurred by the Message Understanding Conferences held in the 1980s and 1990s [50].

A typical IE workflow consists of two core tasks: named entity recognition (NER) and relationship extraction (RE). NER deals with recognizing words or word groups referencing known entities, for example people, organizations, or places [93]. The semantic categories for these so-called named entities are highly domain-specific. For example, protein names are relevant in the biomedical domain [77], whereas product names are important for analyzing user-generated reviews [60]. Dictionaries and machine learning are common approaches for NER, achieving accuracies of up to 95 %, dependent on the domain [40, 59, 75, 144]. Due to the high degree of domain specificity, accuracies for novel types are hard to predict: Approaches or features working well in domain A might fail in domain B.

NER is a prerequisite for further steps in text analyses, such as relationship extraction [8]. RE uncovers semantic relationship between named entities, e.g. "who married whom" or "where did what happen". Common aliases for RE are template filling [21], event extraction [110] or semantic role labeling [99]. The targeted relationships may incorporate two or more entities, termed by *binary* or *n-ary* relationship extraction, respectively. Most research so far focuses on the binary case, where a rich set of methods is available, including pattern-based [46], rule-based [142], or machine learning-based approaches [107]. These methods often utilize linguistic preprocessing, e.g. part-of-speech tagging, stemming, and constituent or dependency parsing [82]. Recent binary RE systems score in the range of 40 % to 80 % [46, 64], with results again being highly task and domain specific.

However, often more than two arguments are required to express real-world facts correctly. For example, the sentence "The Nobel Prize in Chemistry 2014 was awarded jointly to Eric Betzig, Stefan W. Hell and William E. Moerner [...]."[1] uses four arguments to describe the awarding: WHAT, WHAT-SUBCATEGORY, WHEN, and WHOM. When moving to the n-ary case, the extraction process becomes more challenging than for the binary case. First, more entities need to be identified. Given a probability of

[1] http://www.nobelprize.org/nobel_prizes/chemistry/laureates/2014/

$X < 1$ for recognizing one entity, the probability X^n of recognizing n entities—given independence—decreases exponentially with increasing n. This potentially lowers the RE recall, i.e. the portion of extracted among all contained instances, as non-recognized entities induce non- or incompletely extracted relationship tuples. Furthermore, the relationship definition might allow incomplete instances, as texts often miss all desired entities. So the actual number of entities per instance may vary between 1 and n. Moreover, these entities potentially span multiple sentences. Consequently, reported results drop to 46 % correctness for n-ary relationships [108, 109]. However, as entities in the n-ary case usually belong to different categories (person, date, etc.), it might be rare that many equally-categorized entities are found in the same context. This lowers the ambiguity by reducing the number of valid entity combinations and therefore potentially eases n-ary relationship extraction, likely resulting in increased RE precision, i.e. the portion of correct among all extracted instances. In the extreme, the pure co-occurrence of compatible entities within a context might be sufficient to deduce a relationship tuple.

In this chapter, we compare 15 state-of-the-art approaches for extracting n-ary relationships (Section 4.2). They allow extracting arbitrary facts from texts, formalized as n-ary tuples. These approaches are part of the extraction module in our framework. The module takes a set of documents and an information request—encoded as relationship examples—as input, extracts the demanded facts, and forwards them to the subsequent modules (see Figure 1.2 on page 4). We evaluate these methods on a 4-ary relationship on three novel corpora (Section 4.4). Section 4.7 discusses our findings and we conclude with related work (Section 4.8).

4.1 Example Relationship

Before presenting the extraction methods, we define a 4-ary relationship modeling casualty reports as a running example for n-ary relationships. It is used in the remainder of this chapter to give examples and evaluate the methods.

Relief organizations seek for reliable and timely data describing the event and its aftermath. Casualty numbers here are an indicator for the scale of damage, determining the appropriate extend of relief operations and supports their coordination [117].

We formalize reported casualties as 4-tuples [modifier, quantity, subject, type]. The four entities taking part in the relationship are categorized as follows:

- Modifier: modifies quantity values, e.g. 'at least', 'about', or 'more than'.
- Quantity: numbers casualties and consists of two subcategories: cardinal ('12', 'ten', 'no', 'a') and vague ('many', 'hundreds', 'some').
- Subject: characterizes casualties explicitly, e.g. 'people', 'villagers', 'students'.
- Type: describes the type of damage and consists of multiple subcategories: killed ('death toll', 'died'), injured ('wounded', 'broken leg'), trapped, missing ('unaccounted for'), homeless, affected, and evacuated. The type can also be a combination of the given subcategories, e.g. 'killed or listed as missing'.

Moreover, relationship tuples must fulfill the following two constraints for validity:

(1) A type entity is set.

(2) If a modifier entity is set, a quantity entity is set as well.

By analyzing two sentences from a real earthquake report[2]

> "The death toll [...] is now at least 32, with 467 injuries [...]"
>
> "[...] which left about five million people homeless."

we can identify three facts regarding casualties: ≥32 killed, 467 injured, and ≈5m people homeless. Formalizing these facts with the above definition results in three relationship tuples: ['at least', '32', –, 'death toll'], [–, '467', –, 'injuries'], and ['about', 'five million', 'people', 'homeless'].

4.2 Methods

The extraction process we study in this chapter consists of two major steps: named entity recognition (NER, Section 4.2.2) and relationship extraction (RE, Section 4.2.3). The first step recognizes all entities—words or word groups relevant for the targeted relationship, e.g. 'ten', 'more than', or 'injured'. Here we apply two (plus one) interchangeable methods: dictionaries combined with regular expressions and conditional random fields, accompanied by oracles (providing gold standard entities). Based on the recognized entities, the second step extracts relationship instances as combinations of entities, e.g. ['more than', '20', 'people', 'missing']. We present five interchangeable methods for this step, utilizing token distances in sentences, patterns in dependency graphs, and shallow linguistic kernels for classifications with support vector machines.

4.2.1 Preprocessing

To enable fact extraction from a set of documents, we need to prepare these documents in advance to meet the input requirements of the applied methods. For HTML document, we convert the HTML code into text strings, distinguishing between the title, the description and the content. The first two are defined by their respective HTML tag, whereas the actual page content is extracted by Boilerpipe [69], a boilerplate removal library. Boilerplate here denotes non-content elements on web pages, such as advertisements or navigation bars.

Titles, descriptions and first text blocks of news-like articles usually offer key facts playing an important role in information extraction. These article elements often contain particular prefixes and suffixes reporting article metadata. Two examples are (affixes are marked gray):

'NANCHANG, June 9 (Xinhua) -- An elderly couple [...] confirmed dead [...]'[3]

'6.9 Negros quake kills 7: NDRRMC | ABS-CBN News'[4]

[2] http://news.bbc.co.uk/2/hi/asia-pacific/7591152.stm
[3] http://news.xinhuanet.com/english2010/china/2011-06/09/c_13919421.htm
[4] http://www.abs-cbnnews.com/nation/regions/02/06/12/69-negros-quake-kills-7-ndrrmc

These affixes contain toponyms ('NANCHANG'), date expression ('June 9'), abbreviations ('NDRRMC'), proper names ('ABS-CBN News'), or unusual characters ('|'). We observed that those affixes interfere with (deep) linguistic parsing, negatively effecting the extraction results. The reason is that most affixes contain out-of-vocabulary terms, i.e. words or characters unknown to the applied parsing model. Therefore we optionally discard these affixes by heuristic rules, derived from external development data.

Segmenting strings into sentences and words is a crucial prerequisite for further text analyzes, as most methods work on the sentence/word level. Furthermore, POS tags as well as stems are later utilized as important word features (Section 4.2.2 and 4.2.3). Part-of-speech (POS) tagging assigns each word to its grammatical category, such as verb or noun. Stems are artificial roots of words, useful to abstract from concrete word forms. For example, 'example' and 'examples' share the root 'exampl', which could be used to match both variants [70]. Given text documents or transformed HTML, we apply the following (language-dependent) basic natural language processing routines [82]:

Sentence Splitting To detect sentence boundaries, we use the `Sentence Detector`, available as part of the Apache OpenNLP library[5].

Tokenization To detect token boundaries, i.e. words or punctuations, we apply a program called `ewDciTokStrm.C`[6], available as part of the applied dependency parser (Section 4.2.3).

Part-of-Speech Tagging To assign grammatical categories, we apply the POS tagger contained in the OpenNLP library.

Stemming To generate stems, we apply an enhanced version[7] of the Porter stemmer [102], distributed as part of the Apache Lucene project[8].

4.2.2 Named Entity Recognition

Given tokenized and linguistically annotated sentences as input, the extraction process starts by recognizing the targeted entities among the tokens. Entities may consist of a single token ('ten', ' injured') or span multiple tokens ('5 million', 'at least').

Dictionary & Regular Expression

Using this NER approach, all cardinal quantities are recognized by a regular expression, all other entities by a dictionary derived from labeled training data. The component contains two optional post filters for cardinal quantities: M-Filter and A-Filter. As the regular expression recognizing cardinals does not encode the context, the M-Filter removes potential false positives by considering those surrounded by units of measurement, e.g. 'ft', 'km', '$', or '%'. The A-Filter withdraws all 'a'/'an' annotations, as the vast majority of these terms refer to the indefinite article and not to the cardinal 1, resulting in many false positives.

[5] `https://opennlp.apache.org/`
[6] `http://bllip.cs.brown.edu/download/reranking-parserAug06.tar.gz`
[7] `http://snowball.tartarus.org/algorithms/english/stemmer.html`
[8] `https://lucene.apache.org/`

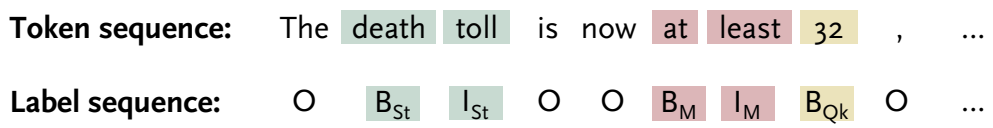


Figure 4.1: A tokenized sentence modeled as sequence of labels following the IOB schema. The label indices correspond to the category of the entity, e.g. `St` for the type 'killed', `M` for modifiers, and `Qk` for cardinal quantities.

Conditional Random Field

The second NER approach is based on Conditional Random Fields. CRFs are probabilistic, graphical models used to label sequential data, such as sentence tokens [72]. They have been successfully applied to entity recognition [87] or part-of-speech tagging [72]. In contrast to (context-free) dictionaries, CRFs encode the context of tokens, enabling superior results. We use a first-order, linear-chain CRF, implemented as part of the MALLET package [86]. To model token sequences as label sequences as required by the CRF, we apply the IOB schema, distinguishing between *I*nner entity tokens, tokens *O*ther than entity tokens, and tokens at the *B*eginning of entities (Figure 4.1). This conversion results in 23 labels, given the number of different (sub-)categories of entities for our example 4-ary relationship (Section 4.1). Each token is represented by its feature vector using standard features, i.e. the textual token value, the stem, and the POS tag. We also utilize the entity dictionary and the regular expression described above to derive features by matching token sequences against the dictionary or regular expression, respectively. Previous results indicated a high recall for these features, emphasizing their potential benefit for entity recognition [29]. As the regular expression also matches year numbers, leading to false positives, we introduce a number range feature. It signals all cardinals between configurable limits as possible year numbers, adding a hint to the model to distinguish between quantities and dates. As the existence of one entity might influences the probability for further entities, we add a sliding window feature, signaling other entity candidates in the surrounding. Again, candidates are defined by matching token sequences against the dictionary and regular expression. Finally, each feature vector is extended by all features of the preceding and subsequent token, so-called offset conjunction [68], incorporating further contextual information.

CRFs are trained using annotated data, i.e. sentences with tokens converted to feature vectors and labeled according to the IOB schema (Figure 4.1). During the training, the CRF "learns" conditional probabilities between IOB labels and token features. During the annotation, the CRF determines the most probably sequence of IOB labels for the tokens to annotate based on the tokens' features and the trained model. Optionally, training data for the CRF can be pre-filtered to contain only sentences actually containing entities, reducing false evidence. In general, CRFs tend to favor precision over recall in NER tasks [90], but recall is important in extraction pipelines, especially in its early stages. Non-recognized entities always lead to non-extracted relationship instances, whereas false-recognized entities might be filtered out in later stages. Reducing false evidence via the Training-Filter is a simple countermeasure in machine learning, leading to more optimistic models in terms of recall. Using such optimistic models potentially returns an increased number of false positives. Therefore, the CRF-based annotation is

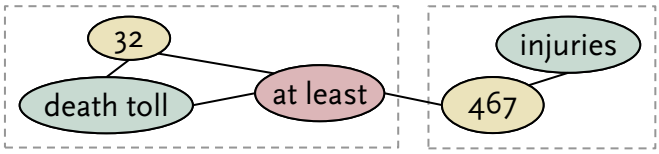


Figure 4.2: An entity graph based on the running example, possibly returned by one of the binary extraction methods. The rectangles mark all contained maximal and valid cliques, i.e. fulfilling all relationship constraints: ⟨'at least', '32', 'death toll'⟩ and ⟨'467', 'injuries'⟩. Note that the clique ⟨'at least', '467'⟩ is not valid though maximal, as it misses the mandatory type entity.

optionally post-processed by a dictionary and regular expression filter, targeting false positives. This WithoutMatch-Filter is the strict equivalent of the similar CRF feature described above. It revokes all annotations from tokens neither contained in the dictionary nor matching the regular expression. Given the observed high recall for the dictionary/regular expression, we expect increased precision and (nearly) even recall after filtering.

Oracle

The third NER approach is a perfect entity recognizer, i.e. an oracle. It is based on gold standard data provided by the user. The oracle approach forms a theoretic but important option in our framework: Although inapplicable for unseen documents, it is very helpful to analyze the effect of error propagation within extraction processes. Having gold standard entity annotations enables us to evaluate the full capabilities of our relationship extraction approaches.

4.2.3 Relationship Extraction

Given recognized entities within a sentence as input, the extraction process continues by inferring semantic relationships between these entities. In our framework, the majority of the RE components follow a two-step approach: First, relationships between pairs of entities are determined, resulting in entity graphs with edges between pairs of related entities (Figure 4.2). By finding maximal cliques in these entity graphs, the binary relationships are merged into tuples of the desired n-ary relationship [88]. This two-step approach has the advantage of enabling the usage of established methods for extracting binary relationships. The extracted relationship tuples are optionally post-filtered by the Enum-filter, targeting at enumerations of facts. Given '[...] killed X and injured Y [...]', our methods potentially extract false tuples, such as [–, 'X', –, 'injured']. As each quantity entity belongs to only one type entity, this Enum-Filter investigates all tuples sharing the same quantity and keeps only the most probable one. For this purpose, the Enum-filter considers distances between entities and linguistic hints, such as 'and'.

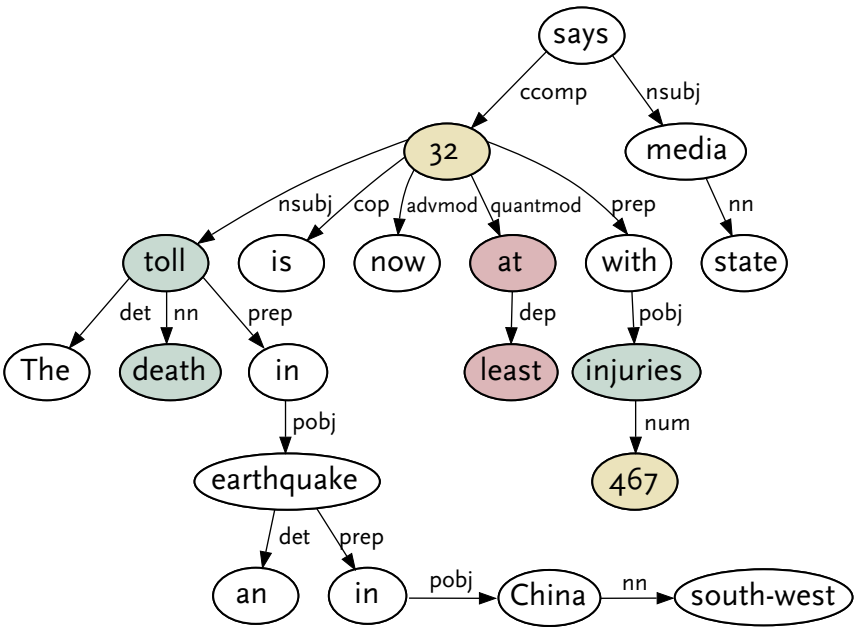


Figure 4.3: The dependency graph for the running example "The death toll in an earthquake in south-west China is now at least 32, with 467 injuries, state media says." The words form the vertices and the dependencies are the typed, directed edges between them, representing the syntactical relationships between the words. For English (and other languages), all words depend (indirectly) on the verb, moving it to the top of the visualized graph.

Token Neighborhood

This co-occurrence-based extraction approach determines a semantic relationship between two entities by their token distance on the sentence level. It combines every entity with all entities of compatible category having a minimal token distances between them, e.g. each modifier with the closest nearby quantity. This method acts as a baseline for more sophisticated relationship extraction methods described below. The resulting entity graphs are further processed by the maximal clique finder.

Pattern Matching in Dependency Graphs

This extraction approach determines a semantic relationship between two entities by matching patterns in dependency graphs. Dependency graphs (Figure 4.3) model the syntactical relationships between the words of a sentence as typed, directed edges between them [82]. By offering direct access to syntactical structures, dependencies often reveal relations between words more easily than if represented as a linear sequence of words, i.e. a sentence [46]. As an example, consider Figure 4.3: The distance on the surface level between the related entities 'death toll' and '32' is ten words, whereas they are directly connected in the corresponding dependency representation.

Dependency graphs are generated by probabilistic parsers [23, 24, 84]. We use the shortest paths between two entities as patterns (Figure 4.4). Bunescu et al. showed

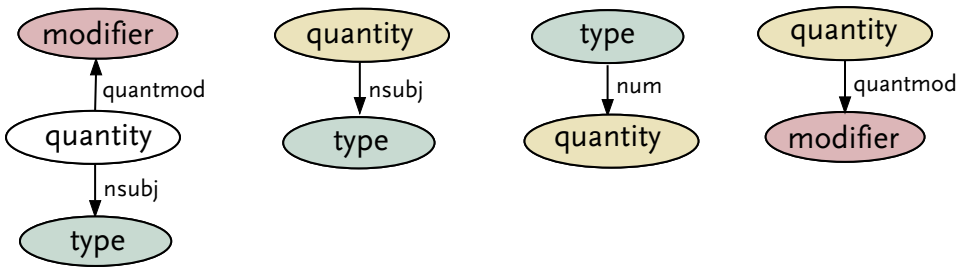


Figure 4.4: Shortest-path patterns, derived from Figure 4.3's dependency graph, encoding vertices with dependency directions and types as match criteria.

that these paths are well suited to capture relationships between entities within sentences [16]. Given labeled training data, all shortest path found in this data constitute the pattern catalog later applied to annotate relationships. During annotation, all possible pairs of entities are matched against all patterns from the catalog with respect to the entity category, the dependency type (edge label), and the dependency direction (edge direction). The resulting entity pairs, i.e. positive matches, are processed by the maximal clique finder.

The pattern matching is adjustable by optionally ignoring the dependency type or direction. These relaxations aim at increasing the recall, as previous results suggest that this method typically delivers low recall but good precision [29].

Classification using Support Vector Machines

The third extraction approach is based on Support Vector Machines (SVMs). SVMs are binary classifiers using a hyperplane in a (potentially) high dimensional vector space as a decision boundary to separate two classes of vectors [26, 52]. Given training data for each class as feature vectors, the SVM calculates the "optimal" separating boundary by calculating the hyperplane whose margin to the nearest vectors of both classes is maximal (Figure 4.5). The term "optimal" means least restrictive, i.e. leaving the maximal margin between the classes to prevent overfitting. The nearest vectors defining the maximal-margin hyperplane are called support vectors, giving the method its name. They are the most difficult instances to classify and provide the most information regarding the location of the decision boundary. During classification, SVMs assign class labels dependent on which side of the hyperplane the new vectors are located. Using the maximal-margin hyperplane has the advantage of inducing the least restriction on the classification of unseen vectors, i.e. instances not contained in the training data.

To find this hyperplane, the two classes have to be linear separable in the original vector space, i.e. their has to exists at least one separating hyperplane. If the two classes are not linear separable in the original vector space, which is mostly the case, they can be transformed into a higher dimensional vector space using a non-linear mapping. Given an appropriate mapping to a sufficient high dimensional space, two classes can always be separated by a hyperplane. This mapping can implicitly be accomplished by applying so-called kernel functions. Calculating the optimal separating hyperplane solely depends on a (artificial) similarity measure between vectors. Kernel functions provide such a

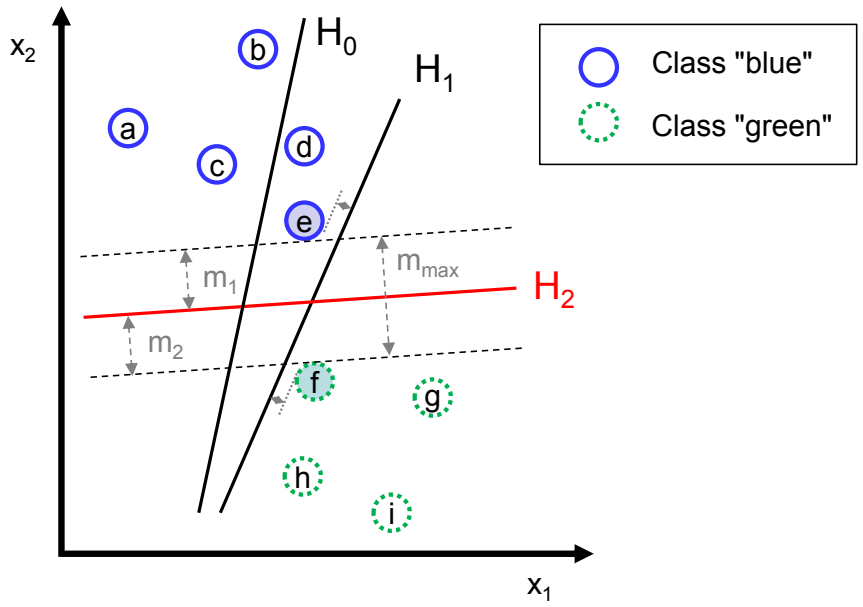


Figure 4.5: This diagram shows how a support vector machine chooses the optimal separating hyperplane for two classes of vectors: $blue = \{a, b, c, d, e\}$ and $green = \{f, g, h, i\}$. H_0 fails to separate the classes. H_1 does separate the classes, but only with a small margin between vectors e and f. H_2 is the optimal separating hyperplane determined by the SVM, as it separates the classes with the maximum margin m_{max} with $m_{max} = m_1 + m_2$ and $m_1 = m_2$. The vectors e and f are the (in this case) two support vectors defining H_2.

similarity measure, transforming vectors non-linearly into a high dimensional space, dependent on the definition of the kernel.

Our SVM-based extraction method determines a semantic relationship between two entities by classifying each possible entity pair utilizing a linguistic kernel function [49]. In contrast to the pattern matching approach, which requires (deep) dependency parsing, this kernel uses only shallow linguistic features beside the token text: stems and POS tags. Tikk et al. showed that this kernel—called jSRE[9]—offers good extraction results compared to other kernels [126]. All positively classified entity pairs are processed by the maximal clique finder.

As we have different categories for entities, we also have different categories for pairs, e.g. <modifier, quantity> or <quantity, subject>. We utilize one SVM per pair category, leading to $\binom{n}{2}$ SVMs for extracting one n-ary relationship (in conjunction with the clique finder).

We allow two relaxations aiming at increasing the recall. These relaxations have to be equally active during training and classification. First, we optionally ignore the direction of entities within pairs defined by the entity categories. Internally, jSRE supports two modes: treating pair elements as equally categorized (undirected) or not equally categorized (directed). For example, dealing with pairs <modifier, quantity>

[9] https://hlt-nlp.fbk.eu/technologies/jsre

in the "directed" mode, which is the default mode, jSRE uses the information which entity is the modifier and which is the quantity. In the optional "undirected" mode, jSRE ignores the additional category information, i.e. treating the entities as equally-categorized, similar to <A,A>. Still, both modes require one SVM per entity pair, but utilize the entity category information differently.

The second relaxation is to ignore the pair category, transforming the classification into a binary-class problem directly applicable to a single SVM. For semantic reasons, the second relaxation is only permissible if the first, i.e. the "undirected" mode, is also enabled.

A third option influences the training stage only, switching between realistic and gold-standard training data. By default, we apply realistic training data for SVMs, i.e. data returned by the selected preceding (imperfect) entity recognizer. Compared to gold-standard data, realistic training data contains false-recognized entities (and misses some entities). These false positives lead to (additional) negative training examples, applied to the SVM. Similar to pre-filtered training data for CRFs, we optionally allow (overoptimistic) gold-standard training data for SVMs, i.e. missing these false positives thus producing less negative training examples. As before, this reduction of false evidence aims at recall increase.

SVM-based Classification with Weighted Cliques

The fourth extraction approach is a variance of the previous one, using the identical SVM-based classification, but subsequently applying an alternative clique finder. This so-called weighted clique finder utilizes the distance from the hyperplane returned by the SVM for each entity pair. The underlying idea is to interpret these distances as confidences and propagate them to cliques instead of focusing on pairs [88]. Large distances are interpreted as strong confidences and small distances as weak confidences. For example, if we found strong positive confidence for pairs <A, B> and <B, C>, and weak negative confidence for <A, C>, we might still want to accept the tuple <A, B, C>. Given the fully connected entity graph with distances as edge weights, the weighted clique finder calculates for each contained clique an average weight. This average is then used to finally decide whether a combination of entities forms an instance of the targeted n-ary relationship or not. In doing so, semantically "weak" non-edges, i.e. pairs classified narrowly negative, can be compensated by "strong" edges, i.e. pairs classified clearly positive (Figure 4.6). By moving the classification from pairs to cliques, this weight-based modification potentially allows more informed decisions about composing relationship tuples.

We limited the use of the weighted clique finder to the SVM-based classification, as this is the only binary relationship extractor evaluated that also returns scores interpretable as confidences.

DARE

We compare our four binary-based extraction approaches with the DARE system, which directly extracts n-ary relationships [138]. DARE determines semantic relationships between entities by matching pattern rules in dependency graphs. In contrast to the

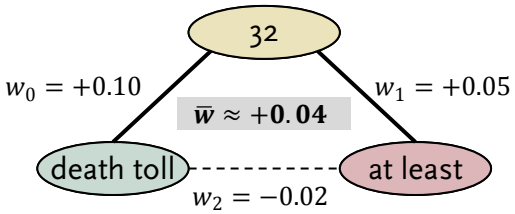
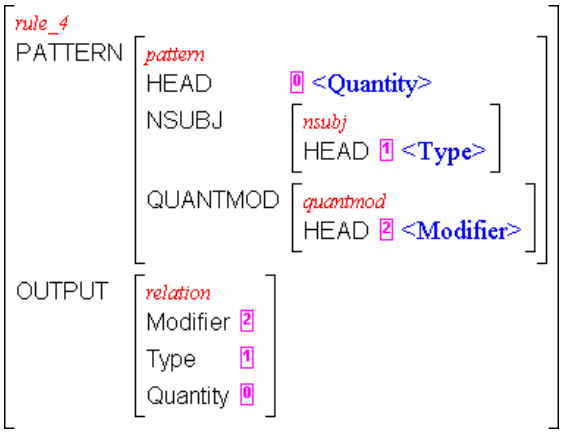
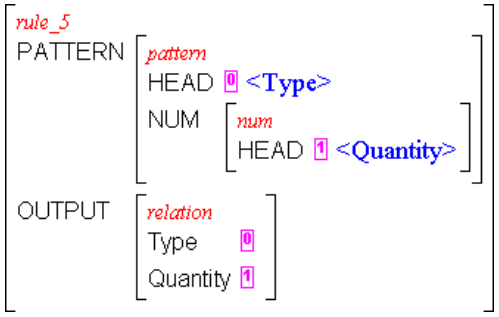


Figure 4.6: Given these entity pairs and their example distances from the hyperplane, noted as edge weights, the SVM would negatively classify <'death toll', 'at least'>, indicated by the dashed line. Consequently, the maximal clique finder would detect the relationship tuple [—, '32', —, 'death toll'] without 'at least'. In contrast, the weighted clique finder would positively classify the complete clique based on the positive mean weight, resulting in ['at least', '32', —, 'death toll']. Note that ['at least', '32', —, —] would be withhold by the maximal clique finder, since the clique ⟨'at least', '32'⟩ is not valid though maximal, as it misses the mandatory type entity.



(a) Rule derived from "The death toll [...] is now at least 32 [...]".



(b) Rule derived from "[...] with 467 injuries [...]".

Figure 4.7: DARE pattern rules examples, derived from Figure 4.3's dependency graph. PATTERN contains the rules enclosed by square brackets, OUTPUT the n-ary tuple matched by the rule. Each rule has a HEAD element, corresponding to a node in the dependency graph. Optionally, it is followed by (sub-)rules, corresponding to connected dependency edges and nodes.

patterns described before, these rules are compositional, potentially referencing other rules and involving more than two entities. They allow connecting entities to pairs, pairs with entities to triples and so on (Figure 4.7). As a consequence, DARE outputs n-ary relationship tuples directly, extending our method collection by a non-clique-based approach. As for the dependency patterns approach (Section 4.2.3), we allow two matching relaxations aiming at recall increase: ignoring the dependency type or direction.

4.3 Document & Relationship Filters

Our framework also contains two configurable, event-independent filters based on the extracted facts. The first filter revokes all documents from further processing containing more than a specified number of relationship tuples. This filter is intended to separate event-specific articles from compilation-like ones, describing more than one event[10]. The second filter allows to withdraw potentially false relationship tuples, i.e. violating user-defined constraints. For instance, knowing in advance the expectable value range of the targeted (numerical) facts allows to define upper/lower limits. Given the example relationship, we might want to filter out all tuples reporting that more than 150 000 people are injured, if we do not expect such high numbers.

4.4 Evaluation

We investigated the suitability of the extraction approaches by comparing their performance at extracting our example 4-ary relationship [modifier, quantity, subject, type] covering casualty reports (Section 4.1) in three corpora (Section 4.5). Each RE method

- Token Neighborhood with maximal clique finder (TN)
- Pattern Matching in dependency graphs with maximal clique finder (PM)
- Support Vector Machine with maximal clique finder (SVM_{MC})
- Support Vector Machine with weighted clique finder (SVM_{WC})
- DARE

needs to be combined with one preceding NER method

- Dictionary & Regular Expression (DctRgx)
- Conditional Random Field (CRF)
- Oracle

to build an extraction pipeline. Figure 4.8 illustrates the possible NER/RE combinations we compare, resulting in 15 relationship extraction approaches. Section 4.6 reports the measured performances for these NER/RE approaches. This includes an

[10] `http://news.bbc.co.uk/2/hi/2059330.stm`

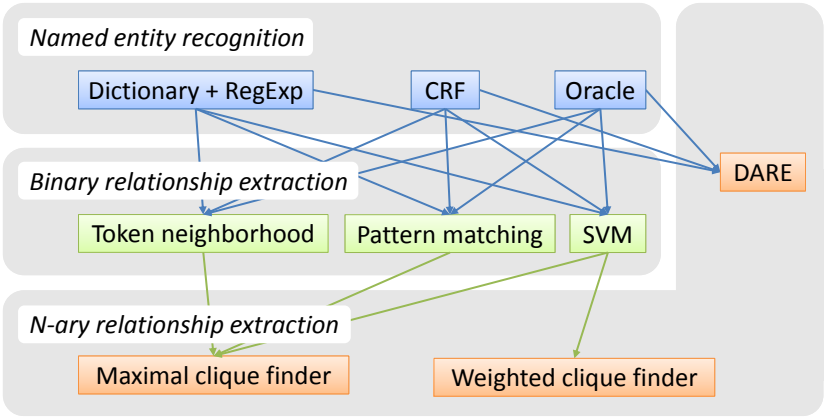


Figure 4.8: Possible combinations of entity recognizers and relationship extractors, together forming a multi-step process to extract n-ary relationships.

examination of the impact of different training data quantities and relationship sizes. We also tested the robustness of learned extraction models across corpora/domains. The purpose of all experiments is to identify the critical issues in n-ary relationship extraction.

4.5 Data Sets

Our evaluation is based on three novel corpora, two of them consisting of news articles and one of Wikipedia articles (Table 4.1). The news articles were manually collected from the web in 2009, 2010, and 2012, reporting on various earthquakes and floods, respectively. The Wikipedia articles were manually collected as well, covering reports on earthquakes that occurred before 2010.

All articles were automatically segmented into sentences and tokens (Section 4.2.1), implicitly checked by annotators during the annotation. Each article was manually annotated with the example 4-ary relationship, covering six casualty categories and their combinations: 'injured', 'killed', 'homeless', 'affected', 'missing', and 'trapped' (Section 4.1). In addition, the flood articles contain 'evacuated' as seventh category.

The annotation guidelines applied are listed in Appendix B. They contain examples for each entity category and describe how entities form n-ary relationship tuples. We developed these guideline using a subset of the documents and trained two annotators[11] accordingly in three sessions. In the first session, we explained the annotation task, presented the guidelines, supervised their annotation of sample sentences, and discussed the results. After that, they annotated 20 documents on their own, solely equipped with the annotation guidelines and 11 annotated documents serving as examples. In the second session, we compared their annotation with our (gold-standard) annotation and discussed the differences. At this stage, we measured an inter-annotator agreement of ≈80 % by calculating the F1-score (see Section 4.6) between their and our annotation. After that, the annotators annotated 30 different documents, compared and discussed

[11] We kindly thank Christoph Fischer and Jirka Lewandowski.

Table 4.1: Corpus statistics; tuple size refers to the number of set entities within relationship tuples.

	Earthquake Wikipedia	Earthquake News	Flood News
Documents	210	245	412
Sentences	7849	4985	8976
containing relationship tuples	8.6 %	21 %	18 %
Tokens	166 476	101 611	193 102
avg(tokens/sentence)*	24.0	22.1	23.2
Relationship tuples	976	1307	2088
type = killed	68 %	64 %	66 %
= injured	17 %	17 %	3.0 %
= trapped	4.1 %	5.7 %	0.34 %
= missing	2.0 %	6.3 %	8.1 %
= homeless	6.0 %	3.9 %	4.2 %
= affected	1.5 %	1.8 %	8.6 %
= evacuated	—	—	9.1 %
= combined	1.5 %	1.6 %	0.67 %
avg(tuples/sentence)*	1.44	1.26	1.27
avg(entities/sentence)*	3.69	3.39	3.52
avg(tokens/entity)	1.14	1.17	1.15
avg(tuple size)	2.79	2.85	2.91
size = 2	39 %	38 %	31 %
= 3	44 %	39 %	46 %
= 4	17 %	23 %	23 %
Positive/negative entity pairs ratio/sentence*	1.34	1.96	2.24
Inter-annotator agreement**	79 %	83 %	81 %

* based on sentences containing at least one tuple
** for untrained annotators solely equipped with the annotation guideline (Appendix B)

```
## A single relationship tuple having all 4 entities set.
## Source: http://news.bbc.co.uk/2/hi/asia-pacific/7591152.stm
Both <O,O> provinces <O,O> were <O,O> severely <O,O> affected <O,O> by <O,O> a <O,O>
devastating <O,O> earthquake <O,O> in <O,O> May <O,O> which <O,O> left <O,O> almost <M> <R2>
70,000 <Qk> <R2> people <O> <R2> dead <St> <R2> . <O,O>

## Two relationship tuples sharing the entities 'many' (vague quantity) and
'people' (subject).
## Source: http://en.wikipedia.org/w/index.php?title=1985_Mexico_City_earthquake
Within <O,O> minutes <O,O> , <O,O> the <O,O> steel-frame <O,O> structure <O,O> collapsed <O,O>
, <O,O> crushing <St> <R19> and <O,O> trapping <Ss> <R20> many <Qv> <R19,R20>
people <O> <R19,R20> inside <O,O> . <O,O>

## This tuple contains the multi-token type entity 'overwhelmed by the waves'.
## Source: http://en.wikipedia.org/w/index.php?title=1783_Calabrian_earthquakes
Many <Qv> <R4> of <O,O> Scilla <O,O> 's <O,O> residents <O> <R4> , <O,O> frightened <O,O>
by <O,O> the <O,O> tremors <O,O> of <O,O> the <O,O> previous <O,O> day <O,O> had <O,O> moved <O,O>
onto <O,O> the <O,O> open <O,O> beach <O,O> for <O,O> the <O,O> night <O,O> , <O,O> where <O,O>
they <O,O> were <O,O> overwhelmed <St> <R4> by <*> <O,O> the <*> <O,O> waves <*> <O,O> . <O,O>
```

Figure 4.9: Example sentences in the annotation format. Each entity is tagged with its category, e.g. **Qk**, and the relationship tuple(s) it is a part of, e.g. **R2**.
Used entity tags: **M** for modifiers, **Qk** for cardinal quantities, **Qv** for vague quantities, **O** for subjects, **St** for damages of the category 'killed', **Ss** for damages of the category 'trapped'.

in the third session. For these documents, we measured an inter-annotator agreement of >90 %. The annotators continued with annotating the remaining documents. The annotation was performed using Notepad++[12], a customizable text editor supporting tag highlighting and auto-completion as well as commenting annotations. Figure 4.9 shows example sentences in the annotation format. Finally, we checked all annotations in dispute and decided on the correct annotation.

The annotation guidelines permit a (rare) fifth entity category, the negation, stating a 5-ary relationship. It is required to correctly annotate negated statements, such as 'These landslides did *not* cause many fatalities [...]'[13]. Overall, we annotated only 9 relationship tuples using this negation entity, 0.20 % of all tuples. We decided to removed all sentences containing one of these tuples, focusing our evaluation on the resulting 4-ary relationship. We also removed sentences containing cross-sentence tuples (0.52 %), i.e. having entities in more than one sentence, and unary tuples (2.7 %), i.e. having only the type entity set. These two classes of tuples are not supported by the evaluated relationship extraction approaches and potentially interfere at the evaluation. Their automatic extraction is out of the scope of this thesis and left for future work. Finally, each corpus is partitioned into a training (2/3) and an evaluation set (1/3) by stratified random sampling on the sentence level.

[12] https://notepad-plus-plus.org/
[13] http://en.wikipedia.org/w/index.php?title=1960_Valdivia_earthquake&oldid=353519137

4.6 Experiments & Results

For each corpus and relationship extraction approach, we determined the most suitable pipeline configuration by 5-fold cross-validating all possible configurations on the training set. Each pipeline configuration consists of one NER component, one RE component, and settings for all switches/filters applicable for these NER/RE components, resulting in 366 configurations to test (per corpus). We maximized the average F1-score (or -measure), the harmonic mean between precision and recall [82]. Precision is the portion of correct results among all returned ones, recall the portion of found results among all contained ones. The year-range feature of the CRF were set between 1000 and 2014, covering reports on historic as well as on recent events. The determined optimal pipelines were trained on the entire training part and evaluated on the evaluation part for each corpus, respectively.

Table 4.2 shows the achieved extraction performances per corpus and RE method, ordered by F1-measure descending, separated by oracle/non-oracle NER. Detailed results and the determined optimal pipeline configurations can be found in Appendix C.

4.6.1 Impact of the Extraction Method

First, we focus on the common scenario of having non-perfect entity annotations. F1-scores rang here from 66 % for Wikipedia articles to 76 % for news reports (Table 4.2). Inspecting the applied (optimized) pipeline configurations reveals that both non-oracle NER approaches as well as all proposed tweaks and filters are utilized to archive optimal extraction results (Table C.2). Comparing the extraction approaches to each other across corpora shows that none of the five methods outperforms the others consistently. Except for TN, all achieve similar results with a F1 drop of at most 5 % compared to the leading method. TN ranks lowest in all corpora, achieving −11 % (avg) compared to the leading method. When switching to oracle NER, we observe a substantial plus of 20 % F1 on average. F1-measures >90 % occur and even TN as simple extraction method becomes competitive.

4.6.2 Impact of the Data Size

We were also interested in the effect of varied training quantities on the extraction performance. These experiments aimed at leveling our different-sized corpora and adding input to the controversy of better methods versus more data [6]. By stratified sampling, we partitioned each training data into bins of sentences containing around 100 tuples. The resulting number of sentences per bin depends on the corpus as the proportion of sentences containing tuples varies (Table 4.1). All relationship extraction methods (except TN) were then reevaluated on the same evaluation data as before, but with an increasing number of training partitions. Figure 4.10 traces the average performances across methods dependent on the training data size. It shows that increasing the training data has a strong positive effect on the extraction performance. For example, the average F1-score for flood reports with non-oracle NER increases from 55 % to 76 % when increasing the size of the training data from ≈100 to ≈1400 tuples. These advancements are mainly caused by gains in recall. Nevertheless, they do not compensate

Table 4.2: Evaluation results per corpus and method, separated by oracle/non-oracle NER and ordered by F1-score. Macro-averages were calculated without token neighborhood. The incorporated NER solution is specified by ^{D} (DctRgx) and ^{C} (CRF).

Earthquake Wikipedia		Earthquake News		Flood News	
Method	F1	Method	F1	Method	F1
Non-oracle NER					
$SVM_{MC}{}^{D}$	.675	PM^{D}	.754	PM^{C}	.783
$SVM_{WC}{}^{C}$	.662	$SVM_{MC}{}^{C}$	.753	$SVM_{WC}{}^{C}$	.764
$DARE^{D}$	.659	$DARE^{D}$	.752	$SVM_{MC}{}^{C}$	.754
PM^{D}	.653	$SVM_{WC}{}^{D}$	.733	$DARE^{C}$	.744
Avg	.662	Avg	.748	Avg	.761
TN^{C}	.575	TN^{C}	.701	TN^{C}	.690
Oracle NER					
SVM_{MC}	.875	SVM_{MC}	.911	SVM_{WC}	.921
SVM_{WC}	.856	PM	.886	PM	.898
PM	.816	SVM_{WC}	.886	SVM_{MC}	.891
DARE	.775	DARE	.866	DARE	.843
Avg	.831	Avg	.887	Avg	.888
TN	.803	TN	.893	TN	.840

DctRgx – Dictionary & Regular Expression
CRF – Conditional Random Field
TN – Token Neighborhood with maximal clique finder
PM – Pattern Matching in Dependency Graphs with maximal clique finder
SVM_{MC} – SVM-based classification with maximal clique finder
SVM_{WC} – SVM-based classification with weighted clique finder
^{D} incorporated DctRgx as NER
^{C} incorporated CRF as NER

the drawbacks of imperfect NER, as non-oracle NER traces exposes no overlaps with oracle ones.

Therefore we investigated the effect of increasing training data on the performance of non-oracle NER. We identified three distinct classes of non-oracle NER configurations in our pipelines, characterized by their precision/recall on the entity level (Table C.2):

- DctRgx providing low precision and high recall,
- CRF providing high precision and medium recall,
- $\text{CRF}_{\text{filtered}}$ (with enabled Training-Filter) providing medium precision and high recall.

We tested one example configuration for each class. Figure 4.11 traces the sole NER performances dependent on the training data size. We find a clear positive effect of increasing training data for both CRF solutions, again mainly due to gains in recall. For example, the $\text{CRF}_{\text{filtered}}$ NER-F1-score for flood reports increases from 65 % to 77 % when increasing the size of the training data from ≈100 to ≈1400 tuples. In contrast, the NER-F1-measure for DctRgx decreases as the drop in precision superposes the recall gain.

4.6.3 Impact of the Tuple Size

We also investigated the dependency between tuples sizes and extraction performances (Figure 4.12). Our results show that larger tuples are much easier to extract, with a F1 increase of 31 % between 2-tuples and 4-tuples on average (non-oracle NER). Given perfect entity annotations, the extraction performance is nearly independent on the tuple size, yielding solely +4 % on large tuples.

4.6.4 Model Robustness across Domains

Most of our NER/RE approaches involve supervised learning, i.e. require annotated training data to learn extraction models. To achieve high quality results, such training data should be sourced from the same domain as the extraction will be applied to. We utilize a broad definition of the term 'domain': it refers to the texts used, especially their type (news article, tweet) or topic (earthquake, flood). In general, generated models from these training data are highly domain-specific as well. While achieving reasonable results in the original domain, they often perform poorly when applied to different, even closely related domains. For instance, Jakob et al. studied recognizing opinion targets in user-generated reviews [60]. They observed a relative F1 decrease of 12 % on average when applying CRF-based models across four topics. Tikk et al. measured the cross-corpus performance of SVM-based models for extracting binary relationships, i.e. protein-protein interactions [127]. Although all corpora consisted of biomedical scientific texts, their experiments revealed a relative F1 decrease of 24 % on average when changing corpora. The general observation for cross-domain experiments is that performance losses correlate with the "closeness" of domains, i.e. their "similarity": "Far away" (less "similar") domains result in greater losses than "close" ("similar")

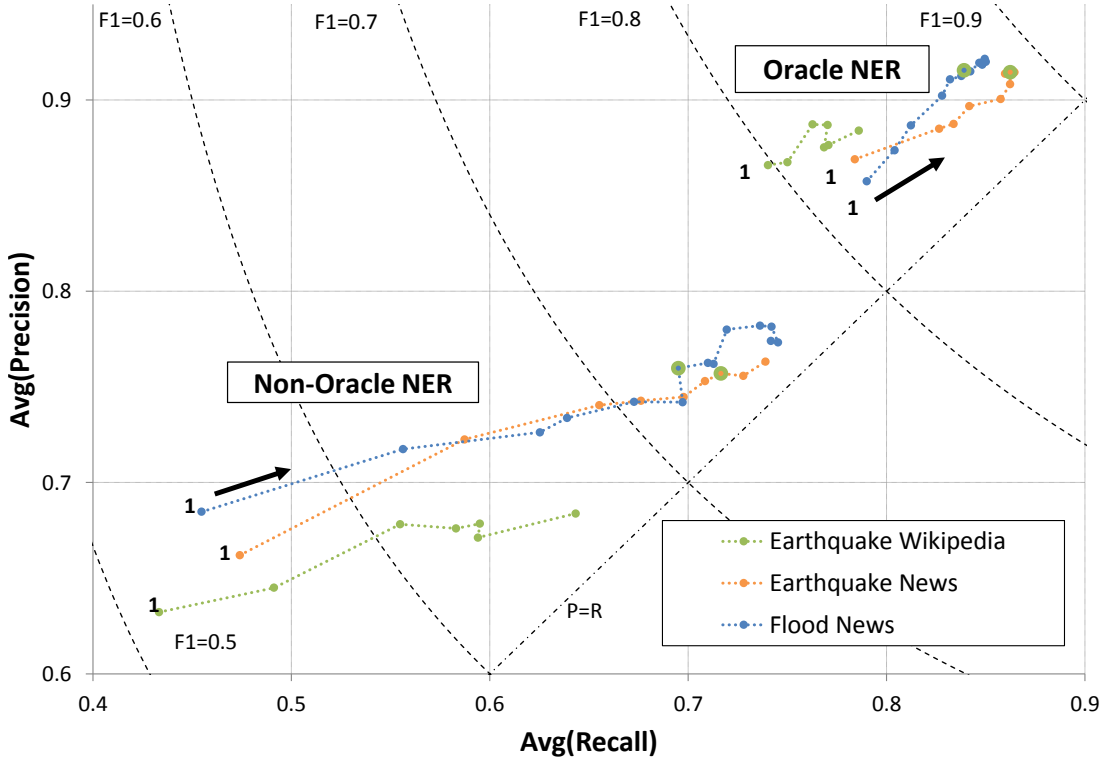


Figure 4.10: Macro-average extraction performance across methods (except token neighborhood), separated by sizes of training data. The left most data point of each trace ('1') corresponds to the results for one partition, containing ≈100 relationship tuples. Each subsequent data point results from another partition added to the training data, each partition containing ≈100 tuples. The black arrows indicate the direction of increasing training data size. The green bordered points within the news traces mark the results for training data comparably sized to the entire Wikipedia training data (≈700 tuples).

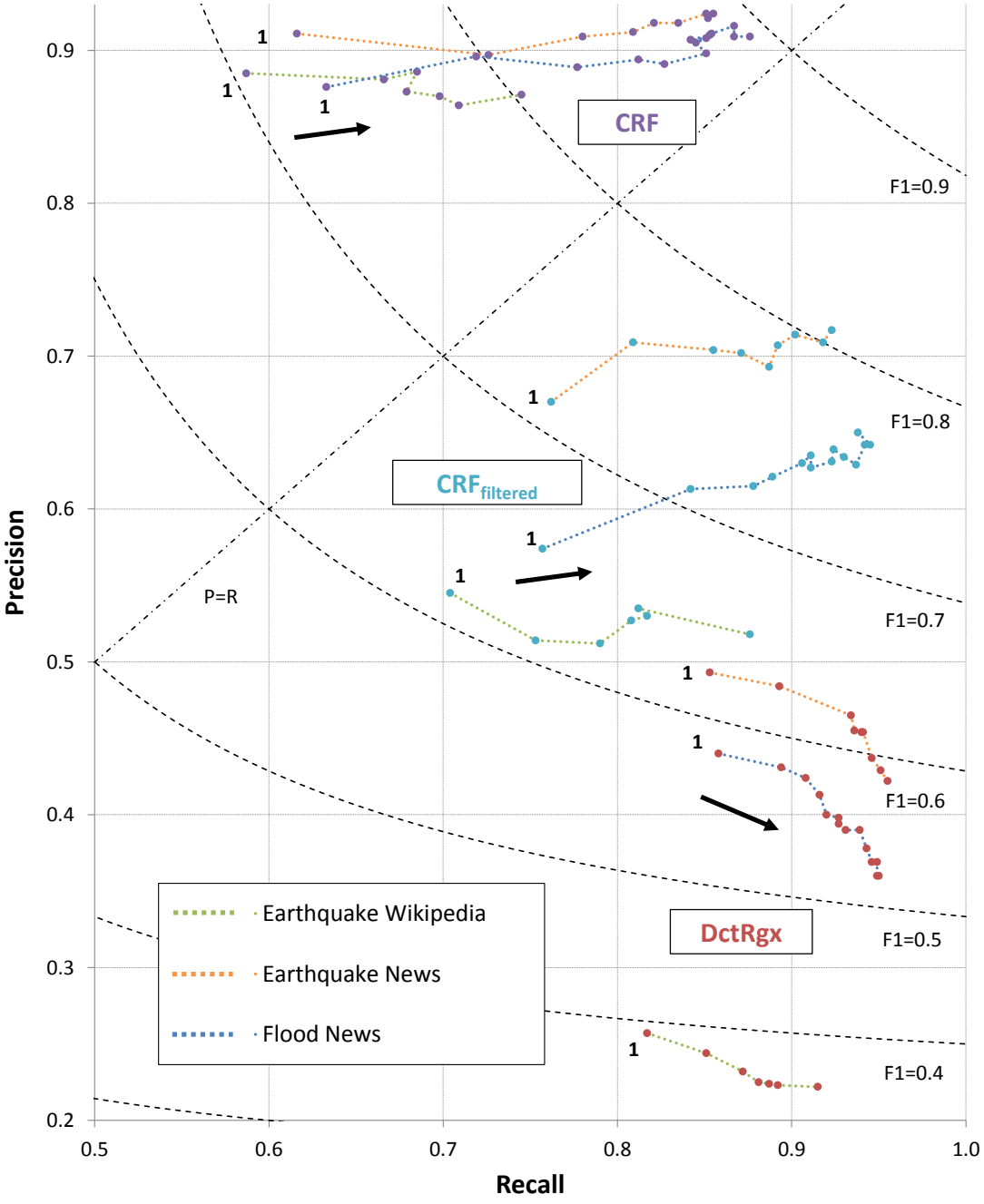


Figure 4.11: NER performances, separated by sizes of training data. The left most data point of each trace ('1') corresponds to the results for one partition, containing ≈100 relationship tuples. Each subsequent data point results from another partition added to the training data, each partition containing ≈100 tuples. The black arrows indicate the direction of increasing training data size.

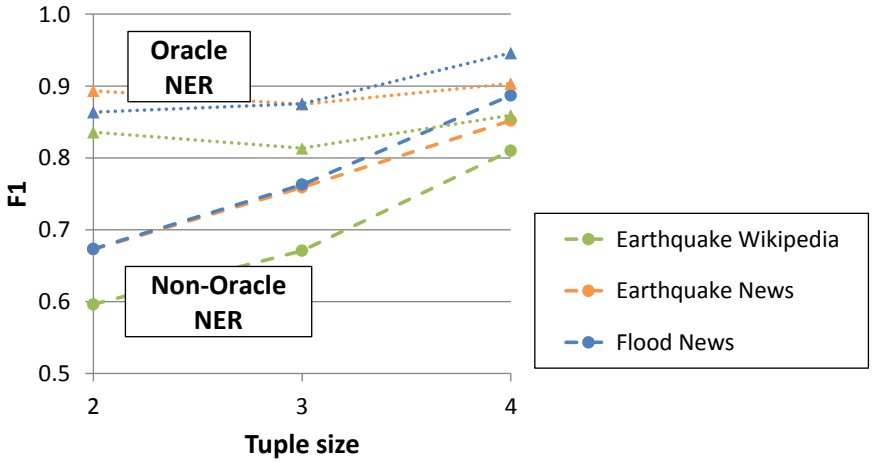


Figure 4.12: Macro-average extraction performance across methods (except token neighborhood), separated by tuple sizes.

domains. An a priori definition for "closeness", i.e. a similarity measure for domains, is difficult to find and potentially depends on the compared domains. Intuitive factors are the language (English, French, etc.), the topic (sport, economy, etc.), or the text type (news article, e-mail, tweet, etc.).

To prevent cross-domain performance losses, information extraction in new domains requires retraining appropriate models. Retraining in turn demands new annotated data, but annotating is an expensive and cumbersome manual task.

An alternative approach is to aim for robust extraction models performing well across domains. These models should enable re-using existing models in new domains, trading little performance losses off against the benefit of saving annotational work. We analyzed the cross-domain suitability of our extraction approaches by testing two—in terms of encoded context thus encoded domain—oppositional extraction models at our three corpora. DctRgx+PM uses little context whereas CRF+SVM$_{\text{MC}}$ incorporates properties of preceding/subsequent tokens considerably. We treat each corpus as a distinct domain, as their documents differ in the event type covered (earthquakes versus floods) and/or the text type used (Wikipedia articles versus news articles). For fair comparison, both news corpora were scaled down to fit the Wikipedia corpus in the number of relationship tuples, retaining tuple distributions in type and size. All models were trained on the training part of the source corpus and testes on the evaluation part of the target corpus, applying identical configurations. Table 4.3 reports the average performance losses when applied across topics (earthquake vs. flood) and text types (Wikipedia vs. news article). The results show that DctRgx+PM models are more robust than CRF+SVM$_{\text{MC}}$ models: losses of $-5.9\,\%$ to $-8.0\,\%$ F1 versus $-8.5\,\%$ to $-17.2\,\%$. Recall and precision declines in all comparisons, the former more than the latter. Losses at topic boundaries exceed those at text type changes and are maximal across both, implying a greater "closeness" between corpora covering the same topic than sharing the same text type.

Table 4.3: Average relative performance alterations for extraction models applied across corpora compared to intra-corpus results. "A ↔ B" means training with corpus A along with evaluation in corpus B and vice versa.

Corpora	Precision		Recall		F1	
	DctRgx +PM	CRF+ SVM_{MC}	DctRgx +PM	CRF+ SVM_{MC}	DctRgx +PM	CRF+ SVM_{MC}
EQ Wiki ↔ EQ news	−1.3 %	−3.3 %	−9.7 %	−12.7 %	−5.9 %	−8.5 %
EQ news ↔ FL news	−2.1 %	−9.8 %	−8.8 %	−15.5 %	−5.8 %	−12.8 %
EQ Wiki ↔ FL news	−3.0 %	−12.8 %	−12.0 %	−21.1 %	−8.0 %	−17.2 %

DctRgx – Dictionary & Regular Expression
CRF – Conditional Random Field
PM – Pattern Matching in Dependency Graphs with maximal clique finder
SVM_{MC} – SVM-based classification with maximal clique finder

4.7 Discussion & Summary

We compared 15 approaches for extracting n-ary relationships from texts. We analyzed their characteristics by evaluating the automatic extraction of a 4-ary relationship modeling casualty reports in three distinct corpora. The approaches achieved F1-scores of up to 78 % on unlabeled texts and up to 92 % with given entities (oracle NER). None of the RE approaches—except TN—considerably outperformed the others, and all achieved comparable results on the same corpus. The large gap between oracle and non-oracle results of +20 % F1 instead strongly indicates that—in our settings—the extraction performance is primarily limited by the preceding NER component. In fact, having more accurate NER would even transform (unsupervised) TN into a competitive RE solution. When comparing the different non-oracle NER approaches, it seems that more precise NER enabled less strict RE and vice versa. For example, best results for SVMs when combined with CRFs were achieved by ignoring the entity pair category and direction, whereas these SVM switches should be disabled when combined with DctRgx.

The measured inter-annotator agreement of around 90 % points out that the analyzed task is not easily solvable, even for trained humans. For example, POS tagging for English texts gave agreements of >96 % [83]. Still, there is a large discrepancy between the automatic and the human performance, but proper NER would narrow the gap.

The analysis of different training data sizes showed the expectable increase in performance by increased data. In fact, the effect of more training data was stronger than the influence of the selected extraction method. This suggests focusing on additional data for further improvements rather than on better relationship extraction methods. Comparing non-oracle NER traces with oracle ones still exposed large performance gaps, again emphasizing accurate NER. De facto, the influence of NER superposed even the effect of increased training data: Analyzing the underlying NER solutions revealed that the observed increases are mainly caused by improvements in NER. Interestingly, the traces also showed that extraction pipelines utilizing DctRgx or $CRF_{filtered}$ might not profit from more training data. These NER approaches already reached a very high recall by only small or even negative advances in precision. It seems that only CRFs can

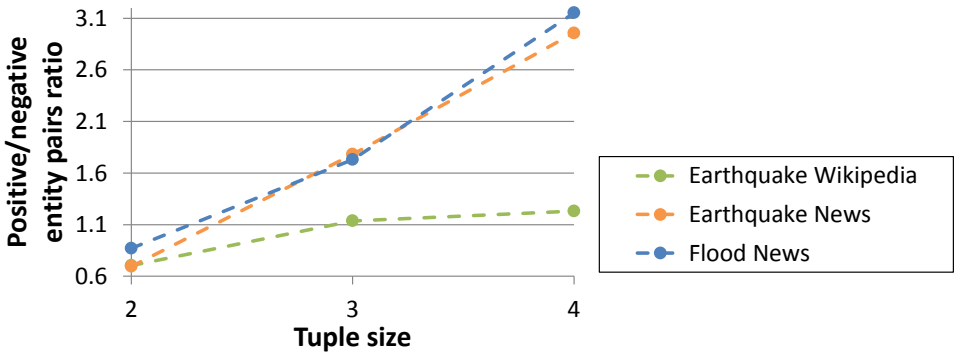


Figure 4.13: Positive/negative entity pairs ratio in sentences containing at least one tuple of the specified size.

benefit from more data, showing constantly high precisions and an increasing recall.

Comparing the extraction performances for different tuple sizes gave the apparently contradictory finding of eased extraction for larger tuples. Note that successfully extracting a 4-ary relationship tuple requires recognizing four entities and $\binom{4}{2} = 6$ entity pairs. Given X^n as the (independent) probability to recognize n entities and $X < 1$, the extraction should become more difficult with increasing n. Of course, entities do not appear independently and given the limited length of sentences, one sentence potentially contains only one complex piece of information, e.g. one 4-ary tuple. For instance, we analyzed the correlation between the tuple size and the positive/negative entity pairs ratio within the tuple's sentence (Figure 4.13). This ratio measures the relation between the number of correct entity pairs and false entity pairs within a sentence, indicating the a priori probability of extracting correct pairs. It depends on the number of entities and respects shared entities (taking part in multiple tuples) as well as entity categories (defining (im)possible pairs). The measured increasing ratios for increasing tuples sizes, e.g. 0.87 at size=2 versus 3.2 at size=4 for flood news, clearly indicate less ambiguity for larger tuples (Figure 4.13). Both results support the hypothesis of less ambiguity and therefore eased information extraction in sentences containing complex information.

Analyzing the cross-domain robustness showed that DctRgx+PM models might be directly applicable to new domains, i.e. without retraining. For example, we measured only 6 % F1 performance drop between news articles for earthquakes and floods. This small decrease might be acceptable in practical applications, especially when offsetting against the costs of acquiring domain-specific training data. Other acceptable cases might be unknown or heterogeneous target domains, e.g. documents covering different (new) event types or including user-generated content. By checking the trained models and comparing the underlying data sets, we identified two main reasons for the observed domain independence; both are connected to each other and equally important. First, sentences reporting on casualties use similar structures and wordings to express facts, independent of the targeted event or text type. Second, most entries of the acquired entity dictionaries and pattern catalogs are comprised of event-type-unspecific words. Comparing earthquake and flood articles, the entity dictionaries overlap by approximately 44 % of their entries. Only about 3 % of all entries are event-type-specific, e.g. 'quake toll' or 'drownings'. These distributions can be observed in the dependency pat-

Table 4.4: Runtime statistics for training extraction models and parsing sentences, separated by method and corpus. All experiments were conducted on a 2012 consumer PC equipped with a 3GHz CPU. Reported relative times are relative to Earthquake News.

	Earthquake Wikipedia	Earthquake News	Flood News
Avg(CRF training time/sentence)	+43.7 %	143 ms	+5.61 %
Avg(SVM training time/sentence)	+103 %	8.50 ms	+26.7 %
Avg(dependency parsing time/sentence)	+8.39 %	361 ms	+9.35 %

terns as well, having an overlap of around 32 % and an event-type specificity of roughly 4 %. For instance, although Figure 4.3's sentence contains the event-type-specific keyword 'earthquake', it is not part of the shortest paths (Figure 4.4).

In all experiments, we observed that information from Wikipedia articles were more difficult to extract than from news articles. This trend held even in the case of oracle NER and even for comparable data sizes (Figure 4.10). We suspect more ambiguity in Wikipedia articles due to smaller tuples and more tuples per sentence (Table 4.1). Both lead to a more disadvantageous positive/negative entity pairs ratio and therefore hindered relationship extraction, also reflected by increased training and parsing times (Table 4.4, Figure 4.13).

4.8 Related Work

Binary Relationship Extraction

A simple approach for relationship extraction is to postulate a semantic connection between relevant entities by mere co-occurrence within a context, usually a sentence [62]. It requires no learning thus no labeled data and—given anaphora resolution [92] and a large enough context—up to 100 % recall is possible. Our Token Neighborhood component uses this approach (Section 4.2.3). However, co-occurrence is also prone to extract false combinations of entities, potentially resulting in a low precision.

These false combinations can be reduced by explicitly defining the context of valid relationship tuples, i.e. rules (or patterns) between entities [53, 71]. For example, the pattern "*<person>* is CEO of *<company>*" might be used to extract the relationship *IsCEO(<person>,<company>)*. Beside the literal token value, rules may utilize further linguistic information, e.g. POS tags or dependency parses [46]. Rule-based extraction systems offer two key features: transparency and customizability. Each match can be explained by the according rule, whereas misses might be avoided in the future by adding/modifying a rule. Pattern-based systems were successfully employed to build large knowledge bases from textual data [76, 122]. But creating and managing the set of rules manually is a laborious task. Fortunately, their acquisition can be automated, given labeled data for the targeted relationship [119]. These labels indicate which entities are related (and which are not). Our pattern matching component uses this supervised approach based on automatically collected patterns from dependency graphs (Section 4.2.3).

(Binary) relationship extraction can also be seen a classification problem: finding all related entity pairs among all existing pairs of entities. Given labeled data, classifiers can be trained by common machine learning techniques [107]. Compared to extraction rules, these classification models lack in transparency and customizability. However, they potentially benefit from hidden correlations in the data hardly realizable by explicit rules. Our SVM component use such an automatic classification approach based on labeled training data (Section 4.2.3).

Obviously, labeled data are a prerequisite for supervised training of extraction models and their evaluation, but its manual creation is a time-consuming task. To reduce these costs, several bootstrapping approaches have been proposed, requiring only a few relationship examples [2, 15]. These seeds are searched in unlabeled data to induce an extraction model which gets re-applied to extract new seeds. Aside from its hard-to-evaluate performance (recall and precision), this semi-supervised approach has several other drawbacks [128]. As the targeted relationship is only defined implicitly by the initial seeds, a semantic drift might take effect between iterations. So the finally extracted relationship potentially differs from the requested one. Also their recall performance strongly depends on redundancy in the data, i.e. seeds occurring in multiple linguistic contexts. One possibility to overcome these limitations is to use many seeds from external data, called distant supervision [73, 91]. Closely connected are unsupervised systems following the so-called Open Information Extraction paradigm [9, 85]. Neglecting any labeled data, they are capable of extracting arbitrary relationships, again hard-to-evaluate and with unknown semantics.

N-ary Relationship Extraction

For the case of n-ary relationship extraction, few comparable previous work exist. McDonald et al. proposed a clique-based approach to synthesize n-ary tuples from the underlying $\binom{n}{2}$ entity pairs [88]. Their approach allows to expand any binary extraction method for the n-ary case and is also applied in our work. They achieved a F1-score of 65 % on extracting a 4-ary relationship in the biomedical domain, given gold-standard entities and classifying pairs by a maximum entropy model. Applied by Afzal to reports on management successions yielded 65 % F1-score as well, based on gold-standard entities and using decision trees as binary classifiers [1]. Wick et al. altered the approach by synthesizing entity clusters based on probabilistic compatibility functions [135]. They achieved a F1-score of 92 % on extracting contact records from student and faculty homepages, using gold-standard entities. Typically, contact records (name, city, phone, email, etc.) are given in a dense and uniformly structured way, potentially easing the task. Xu et al. proposed the dependency pattern-based system DARE following the bootstrapping approach [138]. They yielded 83 % F1-score in extracting a 4-ary relationship reporting on Nobel Price awards using non-oracle NER [137]. Their system was evaluated in Section 4.2.3 and integrated into our framework. Akbik et al. investigated on n-ary relationship extraction in the context of Open IE [3]. They used hand-crafted rules to extract complex facts, operating on the dependency graph level. The evaluation on 500 sentences sampled from the WWW gave 54 % precision, covering up to 6-ary facts. Following the Open IE paradigm, these facts belong to arbitrary relationships and their extraction does not require any preceding NER. More recently, biomedical event extraction received increased attention due to the BioNLP'09 and '11 Shared

Task [66, 67]. The participants applied a multitude of NLP and IE techniques tackling three extraction task of different complexity. The leading teams yielded around 46 % F1-score [108, 109].

5 Information Fusion & Framework Evaluation

Reducing uncertainty by combining information from multiple sources is the goal of information fusion [20]. A prominent example is the position fixing via space-based radionavigation systems, such as GPS, Galileo, or GLONASS. Navigation devices trilaterate signals from multiple satellites to determine the current position [78]. Other examples are using multiple biometric modalities—face, fingerprint, and hand geometry—to improve biometric verifications [111] or the filtering of noisy measurements in wireless sensor networks [94].

In this chapter, we describe methods for time-aware information fusion, i.e. reaching consensus for noisy input data, varying over time. These methods are integrated into the fusion module, the final stage of our framework (see Figure 1.2 on page 4): Its input are event-relevant (Chapter 2) and temporally aligned documents (Chapter 3), providing the demanded facts (Chapter 4), the output are temporally aggregated facts. In this chapter, we use the term 'facts' to describe pieces of information. Facts have a type, related to the information request, and a value, i.e. a name, a number, or a date. For example, if we are interested in the number of injured after a disaster, the fact type is 'number of people injured' and the value could be '10'. In our framework, facts are directly derived from extracted relationship tuples (Chapter 4), i.e. each tuple represent a fact, for example [—, '467', —, 'injuries'] corresponds to fact(type=injured, value=467).

At the fusion stage, incoming documents describe the same event at different points in time, sourced from different publishers. Their contained facts might variate in granularity, correctness, and trustworthiness, representing a potentially inconsistent view on the event as they might be contradictory. We propose time-aware information fusion strategies dealing with these issues, returning reliable, time-resolved views on events.

Regarding (long-lasting) events, the most crucial aspect of the heterogeneity of describing facts is their temporal dimension: Fact values may change within minutes or hours during crisis events, e.g. reported casualty numbers, are up-to-date for years, e.g. who is married to who, or are static, e.g. who invented what. Dependent on the event and fact type, the "truth" evolves over time, but the age of information does not necessarily imply any outdatedness. We also presume a shift in the quantity and probably quality of available information during and after events: In case of disasters, we observed over time a decrease in the number of sources reporting (Figure 5.1), but noticed an increasing validity of the contained information. Incoming facts might be additionally distorted due to errors in preceding modules, i.e. irrelevant documents, wrong temporal alignment, or incorrect fact extraction.

We propose a set of time-aware strategies to deal with such inconsistencies adequately, providing reliable views on events as described by (numerical) facts. First, we reach

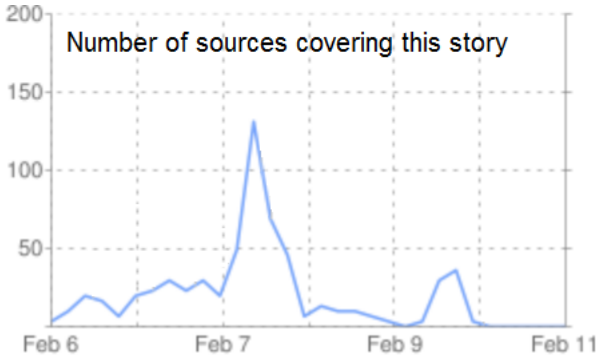


Figure 5.1: "Google News – Timeline of articles" for the Feb 6th Philippines earthquake in 2012 (see Chapter 2); received 2012-02-14.

consensus on the document level utilizing frequencies of extracted facts (Section 5.1). This is complemented by applying a time-aware outlier detector to discard unlikely values (Section 5.2). Finally, we fuse all remaining facts across documents by time-dependent aggregation (Section 5.3). The result is a time-dependent function describing the evolution of the requested (numerical) facts. In Section 5.4, we report the results for two comprehensive case studies applying our framework: Gathering data on multiple earthquakes and floods from the web. We discuss our findings in Section 5.5 and conclude with related work (Section 5.6).

5.1 Intra-Document Fusion

The first step in our fusion application to time-aware data is to aggregate for each document all extracted facts of similar type into one value, i.e. intra-document fusion. The underlying hypothesis is that each document, even if it reports different values for the same fact type, can be reduced to one most-likely or most-desired value. Articles reporting on disasters might contain the officially announced casualty number, an estimate from on-site units, reports on local casualties, or historical numbers from previous events in this region[1]. For instance, our news corpora in Chapter 4 quote an average of 3.6 tuples per document[2] for relationship tuples of type 'killed' having a cardinal quantity, even without including the document's title or description (see below). All numbers are extracted and relevant, but in most cases the desired information is the official count. We utilize that generally key information in news-like articles are contained in the title, the description, or the first sentence/paragraph of the content [134]. Also, key facts are often repeated across these article elements. Hence, we propagate for each fact type the most frequent value or, if ambiguous, the first value as the fusion result on the document level for news-like articles. This frequency and order-based strategy is independent on the fact type and supports non-numerical facts as well, e.g. nominal or ordinal data. Other selectable aggregates are the first or the last value, and the minimum, the maximum, or the median for at least ordinal values. The output of the intra-document fusion is exactly one fact per document (per fact type).

[1] http://www.bbc.co.uk/news/world-asia-16901385
[2] based on documents containing at least one tuple of this kind

5.2 Outlier Detection & Removal

The next step is to detect among all facts aggregated at document-level, i.e. one fact per document (per type), those which are hardly correct, called outliers. These would distract subsequent aggregate functions, for instance curve fitting (Section 5.3) and shall therefore be excluded from further processing. We achieve this by comparing current facts with facts previous in time, resulting in an adapted version of an online median filter [89]. Median filters in general have the advantage of not assuming any specific probability distribution of the values, supporting even discontinuities.

Ordering the documents according to their publication date creates a temporal sequence of facts, thus a sequence of their (numerical) values. Starting with the oldest fact in the sequence, we calculate for each fact the median of the last n fact values (inclusive), i.e. a sliding median with the window size n. We mark the examined fact as outlier if its value is below a fraction or above a multiple of that median, else as inlier. These thresholds might be time-dependent, allowing to narrow the inlier range as time elapses.

In contrast to traditional median filters, we do not replace detected outliers but keep them in the temporal sequence of facts. In doing so, they will be included in later median calculations, permitting considerable but valid jumps in the value sequence. For example, casualty numbers tend to alter in large degrees, notably in the early stage of disasters. The drawback of keeping all outliers in the sequence, especially far away ones, is that they potentially expand the inlier range to much, diminishing the filter effect. We therefore define a second pair of thresholds to capture far away and thus very certain outliers, which will be excluded from later median calculations. We call kept outliers 'outliers-to-memorize' and excluded outliers 'outliers-to-ignore'. All inliers are returned by the detector for further processing in the framework, outliers-to-memorize are internally memorized, but not returned, and outliers-to-ignore are neither memorized nor returned. Accordingly, the output of the Outlier detector is a subset of its input: all those document-level facts classified as inliers.

The window size n and suitable thresholds for both types of outliers should be derived from training data. Choosing the appropriate windows size depends on the expected density of outliers in the sequence. If n is too small, more than $n/2$ consecutive outliers will dominate the median, classifying true inliers as outliers and vice versa. If n is too large, legal jumps in the value sequence will be misclassified as outliers.

Recognizing "unlikely" values among non-numerical facts, e.g. nominal or ordinal data, is highly type-specific and the concept of outliers may not apply at all.

5.3 Inter-Document Fusion

The final step in our fusion module is to aggregate all inliers over time, i.e. aggregating facts across documents with respect to their timestamp. The purpose of this inter-document fusion is two-fold: minimizing the effect of erroneous facts and disambiguating facts having the same timestamp. If the majority of documents within a time range quote the fact value X and only a few the value Y ($\neq$X), we might want to favor X, supposing redundancy on the web as confidence indicator. In doing so, we mitigate the

effect of incorrect facts introduced by erroneous processing in our framework or false reports ("fake news"). We also need to define the distinct fact value returned at t, if we have multiple inliers (documents) with the timestamp t. The output of the final fusion stage are functions describing the evolutions of facts over time, i.e. returning exactly one fact value for all points in time t with $t \geq t_0$.

For this inter-document fusion, we propose to use curve fitting with specific classes of functions, based on the targeted event type and fact type. We assume that selecting suitable functions to encode typical evolutions of facts leads to advantageous fusing of noisy inliers. For instance, if we know that the targeted event type and fact value typically shows a linear increase over time, we should apply this a priori knowledge and fit a linear function. These suitable functions have to be given by experts or determined by training data. Our fusion module supports various function families, each defined by an expression containing a number of parameters, e.g. the affine linear family $f(x) = a \cdot x + b$ with parameters a, b. The parameter values are optimized based on the inlier values, utilizing a least-squares approach [48]. The optimization process can be influenced by assigning configurable weights to inliers affecting the fitting error calculation. For example, confident (or young) values might be assigned weights above the default and doubtful (or old) values weights below. This time-dependent weighting allows to favor young and probably more up-to-date facts without neglecting old information. Suitable weights should be derived from training data. The resulting parametrized functions are directly applicable to return exactly one value per point in time as requested by the fusion module.

Additionally, the module includes configurable sliding window aggregates, such as median or mean. Their aggregation of the last n inliers forms a step function, returning exactly one value per point in time as well. The appropriate window size n should be derived from training data.

Aggregating non-numerical facts across documents, e.g. nominal or ordinal data, is again highly type-specific. Possible approaches are using value frequencies within sliding windows, e.g. majority voting.

5.4 Evaluation

Assessing the suitability of our proposed fusion approaches requires realistic, real-world, noisy data as input, describing real-world events. The best way of producing such data is to apply our entire framework for real-world events, incorporating the fusion module as the final stage.

We evaluated the fusion module within the *entire* framework in two comprehensive case studies: tracing the number of casualties reported after earthquakes and after floods. Both are examples for long-lasting events with crucial changes in the available information. Relief organizations seek for reliable and timely data describing the event and its aftermath. Casualty numbers here are an indicator for the scale of damage, determining the appropriate extend of relief operations and supports their coordination [117]. Beside real-world input data, we also require time series of the requested facts acting as evaluation references, i.e. gold standards. Due to the availability of such series, we decided to use past events whose temporal evolution of facts are listed

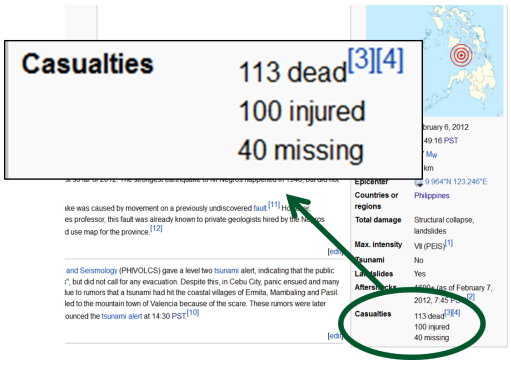


Figure 5.2: Example Wikipedia infobox used to create the gold standard.

on Wikipedia[3], a manually curated online encyclopedia (Section 5.4.1). Section 5.4.2 reports the framework configuration later applied as derived from training data. Our experiments focused on two scenarios: The real-time scenario covers the quality of current fact values returned by the framework whereas the forecast scenario deals with the quality of future values (Section 5.4.3). Results for both scenarios are reported in Section 5.4.4, including measurements of the influence of the different fusion stages on the final result.

5.4.1 Data Sets

Gold Standard

We used Wikipedia articles—more precisely their infoboxes—as reference data as the stored revisions allow accessing previous article versions. These revisions represent descriptions of the same event at different points in time, forming an well-suited gold standard for analyses of time-dependent facts. Wikipedia articles represent a free and comprehensive information source, manually curated by volunteers. Although editable by anybody, article changes are examined by other community members, sometimes even requiring approval. This collaborative approach leads to vandalism removal within minutes [129], ensuring reliability of the presented information. In case of earthquake and flood articles, published information usually do not originate from on-site volunteers. Instead they originate from other sources, e.g. online new papers read by the Wikipedia editors. Therefore, these Wikipedia articles are based on the same information as our framework utilizes: online articles about the event. The difference is that, for Wikipedia, these online articles are manually analyzed and judged, whereas they are automatically processed in our framework.

Focusing on events between 2006 and 2012, we retrieved all article revisions for 45 earthquakes[4] and 26 floods[5]. We used the English Wikipedia, as it offers the largest number of relevant articles and highest update rate among all available languages. After downloading, we automatically extracted the casualty numbers contained in the infoboxes. These infoboxes summarize articles using semi-structured key-value pairs, allowing exact parsing (Figure 5.2). After that, we semi-automatically removed obvi-

[3] https://www.wikipedia.org
[4] https://en.wikipedia.org/wiki/List_of_21st-century_earthquakes
[5] https://en.wikipedia.org/wiki/List_of_floods#21st_century

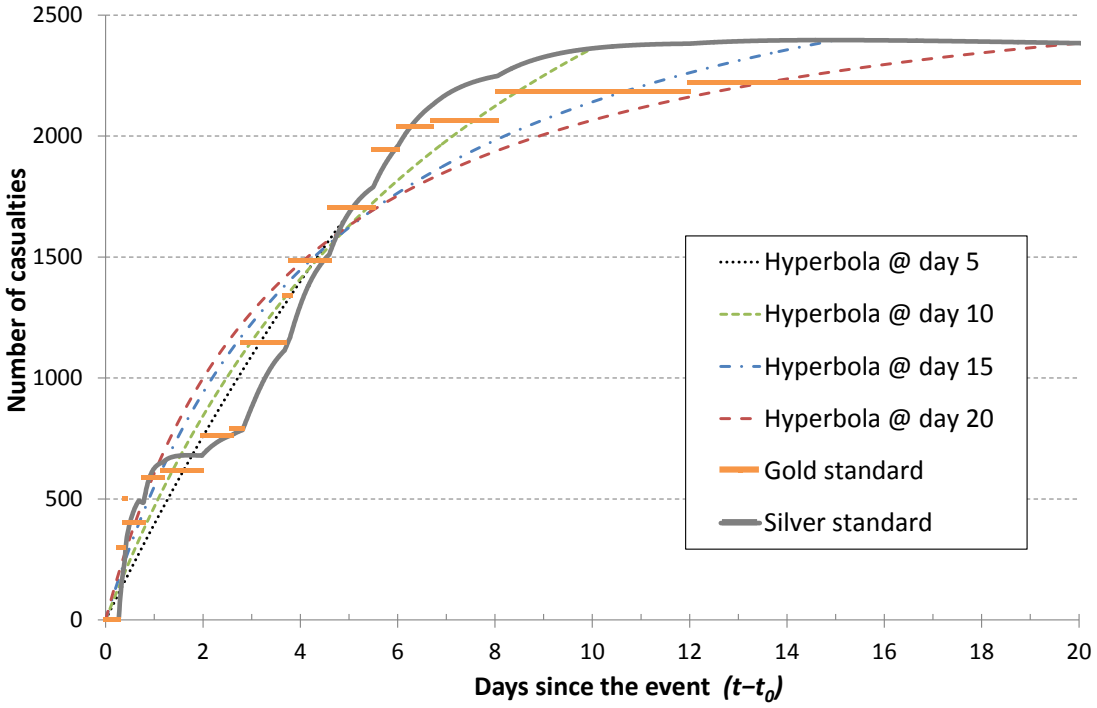


Figure 5.3: The gold standard for the 2010 Yushu earthquake*, sourced from the Wikipedia infoboxes, and the corresponding silver standard. The gold standard is a step function, the silver standard at t is a Monod's hyperbola, fitted to the gold standard at the interval $[t_0, t]$ with t_0 being the event's start date. Example hyperbolas used to calculate the silver standard are plotted for fittings at day $\{5, 10, 15, 20\}$.

*https://en.wikipedia.org/wiki/2010_Yushu_earthquake (part of the training data)

ously incorrect values, e.g. due to vandalism. The extracted time-series of values form step functions and build our *gold standard* (Figure 5.3). From all fetched Wikipedia articles, we selected 33 earthquakes and 9 floods as final data set, listed in Appendix D. In addition, Table 5.5 on page 80 compares key characteristics of the two event types. The selection criteria were: $\geq$ 10 casualties and $\geq$ 5 casualties updates. Ten earthquakes were assigned to the training set by stratified sampling and all remaining events $(23 + 9)$ to the evaluation set. Based on the training set, we determined a suitable framework configuration (Section 5.4.2), later applied in all evaluations.

Start Date t_0

Evaluating time series data also requires to define the point in time where each event started, i.e. t_0. For earthquakes, we use the date provided by the Wikipedia article as the beginning of the event, offering a granularity up to seconds. For floods, determining t_0 is more difficult as floods have no precise start date. We therefore assigned beginnings of floods manually based on the beginning of the causal rainfalls as reported in the media.

Silver Standard

We additionally approximated our gold standard by fitting a continuous function, utilizing a least-squares approach. This approximation carries two potential advantages over the gold standard: First, it represents intermediate values not mentioned on Wikipedia. Second, it reduces the bias induced into the gold standard by incorrect values missed during the semi-automatically cleansing. For both event types, we found that evolutions of casualty numbers are well approximated by saturation functions [106]. Across all events and various days elapsed, the most robust overall fittings are achieved by Monod's hyperbola $f(t) = a \cdot (t - t_0) \cdot (b + t - t_0)^{-1}$. t is the current time here with $t \geq t_0$, $(t - t_0)$ the time elapsed since the beginning of the event, and a, b are the parameters to be optimized.

Fitting these parameters requires to define the fitting interval. The left endpoint is naturally defined by t_0, whereas the definition of a right endpoint is more difficult. Using a fixed right endpoint, e.g. t_0+10 d, would give a static approximation, i.e. independent in t, but event durations vary. Instead, we decided to apply the time elapsed as right endpoint, resulting in the variable interval $[t_0, t]$. The parameters a, b are therefore determined by fitting a hyperbola to the gold standard at the interval $[t_0, t]$, i.e. they depend on t. We call this approximation *silver standard* and report evaluation results for both standards compared to the framework's output. Figure 5.3 plots the two references for an example event.

5.4.2 Framework Configuration

Earthquake Configuration

We used the training set to select appropriate framework modules and to tune their parameters. The objective was to minimize the difference between the framework's output and both references, the gold and the silver standard. More precisely, we tried to minimize the error $e(t)$ applied in the real-time evaluation scenario (Section 5.4.3). We determined the following module configuration by systematically testing different points in time t with $t > t_0$:

Retrival Module

Query Generator Applied queries are based on the *type* terms {'earthquake', 'quake'}, automatically extracting toponyms from the event's Wikipedia article as *location* terms, and automatic *date* terms {<year>, <name of the month>, <name of the weekday>} (Section 2.1).

Search Provider Bing API

Filters We apply several low-pass filters, each described in Section 2.2.

- **Rank** We ignore search results ranking beyond 100, presuming decreasing relevance.
- **Blacklist** We ignore results from Wikipedia, the source of our (reference) gold standard.

Host-only Enabled.

Non-unique We drop results returned for at least five different earthquakes, presuming irrelevance to all earthquakes.

Size We skip documents larger than 500 kB.

Word We revoke documents missing the (sub)string 'quake' within content parts.

Temporal Alignment Module

Alignment We apply PcDE (Section 3.1). Calls to PcDE were parametrized with the page content and the URL, the fetch date, and UTC as default time zone.

Filters We apply a high-pass publication date filter (Section 3.4), targeting at the time delay between an earthquake and possible mentions on the web. It revokes documents dated before t_0+15 min, i.e. probably too early to be valid, since we do not expect informative articles so quickly after earthquakes.

Extraction Module

Preprocessing As described in Section 4.2.1, with Sentence-Affix-Remover enabled.

Extraction We use the NER/RE components which performed best on earthquake news articles, i.e. dictionary+regexp (DctRgx, Section 4.2.2) combined with pattern matching in dependency graphs (PM, Section 4.2.3). Configuration details can be found in Appendix C (Table C.2). All components were trained on the entire earthquake news corpus (Section 4.5).

Filters We apply a low-pass number-of-tuples filter, targeting at compilation-like articles (Section 4.3). It skips documents containing more than 25 relationship tuples. Furthermore, we filter out tuples reporting casualties outside of the interval $[2, 100000]$, as we do not expect more than 100 000 casualties.

All relationship tuples of type 'killed' having a cardinal quantity set are finally propagated to the fusion module as facts of type 'killed'. Their fact values are defined by the extracted cardinal quantity, for instance ['at least', '32', —, 'death toll'] → fact(type=killed, value=32).

Fusion Module

Intra-Document We apply the default strategy, i.e. propagating the most frequent or first fact value (Section 5.1). For the training set, the average number of relevant facts per document[6] was 3.5 (Table 5.1).

Outlier Detection We found a window size of $n = 9$ to be appropriate for our median-based detector and tuned the outlier thresholds accordingly (Section 5.2). All values outside the interval $[0.5 \cdot median, 2.0 \cdot median]$ are labeled as outliers-to-memorize. Moreover, values outside $[0.25 \cdot median, 4.0 \cdot median]$ are excluded

[6] based on documents containing at least one relevant fact

from later median calculations (outliers-to-ignore), if the sequence up to the questioned value has more than five entries and the time elapsed since t_0 is $\geq$24 h. These additional constraints exist to meet the requirements of the vague nature of information in the early stages of (disastrous) events. Since we expect less noisy data as times elapses, we narrow the threshold for outliers-to-ignore down to $[0.30 \cdot median, 3.5 \cdot median]$ after five days and $[0.35 \cdot median, 3.0 \cdot median]$ after ten days. These parameter values are robust in terms of small changes cause little differences at the final error rate due to the subsequent inter-document aggregation.

Inter-Document We tested several function classes including saturation functions as curve-fitting based fusion strategy (Section 5.3) and found that the most robust results are achieved with Monod's hyperbola $f(t) = a \cdot (t - t_0) \cdot (b + t - t_0)^{-1}$. t is current time here with $t \geq t_0$, $(t - t_0)$ the time elapsed since the beginning of the event, and a, b are the parameters to be optimized. Since we presume that reported facts are becoming more trustful with time elapsed, we assign for each value to fit a weight proportional to its age, penalizing fitting errors in the early stages of events less than at the end. This also aims at compensating the presumed decreasing quantity of reported facts over time, where the majority of early data points would dominated the minority of current data points during fitting. To determine if curve fitting is an advantageous inter-document fusion strategy, we define a second, more simple strategy as basis: a sliding mean with a window size of $n = 5$.

Flood Configuration

Given the low number of suitable flood events at Wikipedia, i.e. meeting our criteria, we decided to assign all of them to the evaluation set. Consequently, we reuse the earthquake configuration for those modules missing any training data, e.g. the fusion module.

Retrival Module Same as for earthquakes, except that the query generator uses {'floods', 'flooding'} as *type* terms and the word filter enforces the (sub)string 'flood'.

Temporal Alignment Module Same as for earthquakes.

Extraction Module Same as for earthquakes, except that the NER component is a CRF (Section 4.2.2), as it performed best on flood news articles. Configuration details can be found in Appendix C (Table C.2). The extraction components were trained on the entire flood news corpus (Section 4.5).

Fusion Module Same as for earthquakes.

5.4.3 Experiments

We processed each of the events in our evaluation set with the framework, configured accordingly to the event type (Section 5.4.2). The query generator created up to 100

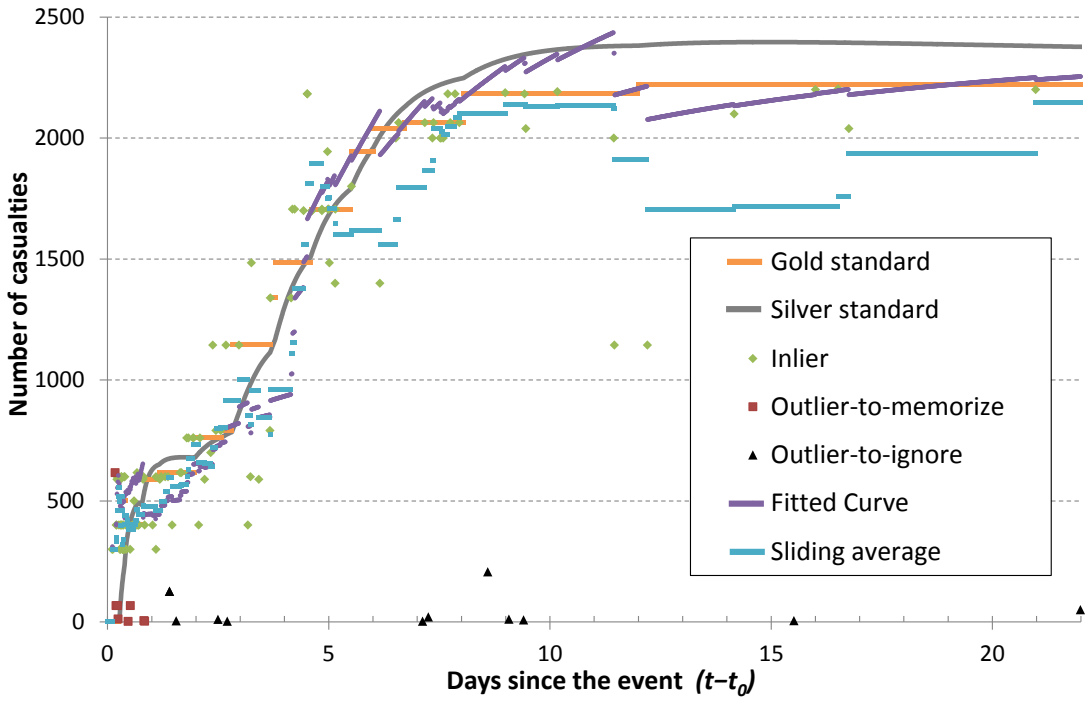


Figure 5.4: The framework's output for the 2010 Yushu earthquake*, a part of the training data. All data points, i.e. inliers, outliers-to-memorize, and outliers-to-ignore, are the result of the intra-document fusion (Section 5.1) in conjunction with the outlier detection (Section 5.2). Five more outliers-to-ignore, $(0.27, 80000)$, $(0.36, 78000)$, $(1.12, 80000)$, $(11.52, 10000)$, and $(16.96, 88000)$, are omitted for better readability. The fitted curve and the sliding average are the result of the inter-document fusion (Section 5.3), applied to all inliers at the interval $[t_0, t]$ with t_0 being the event's start date.

*https://en.wikipedia.org/wiki/2010_Yushu_earthquake

queries per earthquake and 200 per flood event. Searching and downloading the results was conducted in April 2013 for earthquakes and in November 2014 for floods, resulting in 1626 documents per event on average (Table 5.1). After processing and filtering by the retrieval, alignment, and extraction module, the average number of documents for the time span of 33 days elapsed since the event start decreased to 153, forwarded to the fusion module. At the intra-document fusion, the average number of relationship tuples of type 'killed' having a cardinal quantity was 3.9 per document[7]. Outlier detection resulted in 64 inliers and 35 outliers per event on average, the remaining documents contained zero relevant relationship tuples. The inliers are the valid data points to be processed by the inter-document fusion strategy, the final stage of the fusion process.

Figure 5.4 depicts the framework's output for a training event. Based on these documents, our experiments to assess the suitability of the proposed framework focused on two scenarios:

[7]based on documents containing at least one tuple of this kind

Table 5.1: Average numbers per set of events; set sizes given in brackets. The intra-document heterogeneity refers to the number of relevant facts contained.

	Earthquake		Flood
	Training (10)	Evaluation (23)	Evaluation (9)
Unique search results	1975	1576	1952
Downloaded documents	1787	1464	1860
Processed documents	161	142	181
Intra-document heterogeneity	3.5	4.1	3.3
Inliers	69	56	83
Outliers	34	31	45
Ratio(inliers/outliers)	2.2	3.3	4.1

Real-time Scenario The first scenario is the real-time scenario: How well does the framework report the current fact value, given all facts prior in time? In general: Given all documents/extracted facts until a point in time t with $t > t_0$, what quality can be expected for $f(t)$, the framework's output at t? We measure this quality at t by calculating $e_{gold}(t)$, the unsigned relative error between our gold standard $g(t)$ and the framework's output $f(t)$: $e_{gold}(t) = |g(t) - f(t)| \cdot g(t)^{-1}$. Analogously, we define $e_{silver}(t) = |s(t) - f(t)| \cdot s(t)^{-1}$ as the error of $f(t)$ compared to the silver standard $s(t)$. We aggregate e_{gold} and e_{silver} by mean, resulting in the combined error $e(t) = (e_{gold}(t) + e_{silver}(t)) \cdot 0.5$.

Forecast Scenario The second scenario we evaluated is the forecast scenario: How well does the framework predict future fact values, given all facts until now? In general: Given all documents/extracted facts until a point in time t with $t > t_0$, what quality can be expected for $f(t + x)$, the framework's output for $t + x$, with increasing x? We measure this quality at $t + x$ by calculating the error $e(t + x)$ as previously defined. We applied the same documents and framework configuration as in the real-time scenario.

Beside comparing curve fitting and sliding averages as (default) inter-document fusion strategies, we also examined the influence of the other fusion stages on the final result. We evaluated three additional baselines (Figure 5.5):

"Last inlier" bypasses the inter-document fusion and propagates the last inlier returned by the outlier detector.

"Last intra" bypasses the outlier removal as well, propagating solely the result of the intra-document fusion.

"With outliers" bypasses the outlier detector, but applies both inter-document fusion strategies.

We decided to skip baselines bypassing the intra-document fusion as we are interested in summarized numbers of casualties, which should be only one per document.

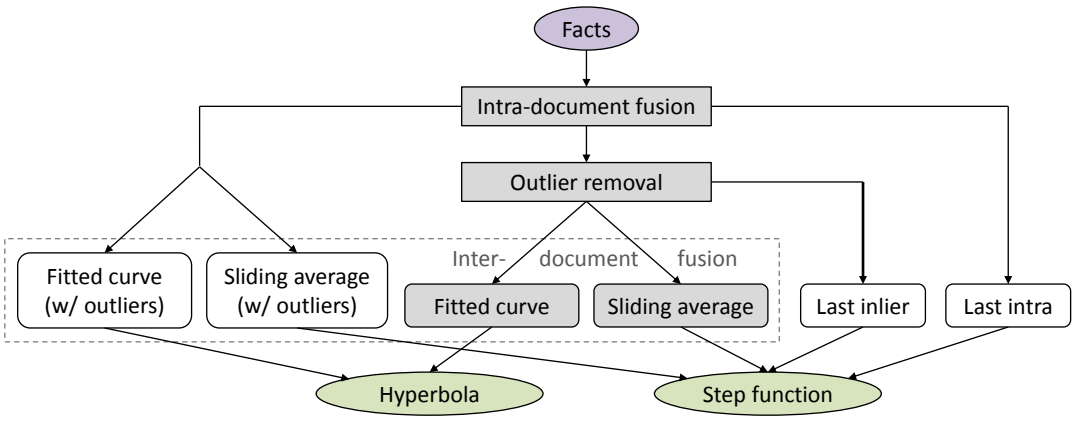


Figure 5.5: Evaluated combinations of fusion stages. Gray boxes mark the (default) fusion strategies, white boxes are the additional baselines.

For each fusion strategy, including the baselines, we calculated the error $e(t)$ for different points in time t with $t > t_0$ for both event types on the respective evaluation set. For the real-time scenario, we started t at t_0+0.5 d for earthquakes, increasing t by 0.25 d until day 7. For floods, we started at $t = t_0$+2 d, increasing t by 1 d until day 30. These maxima are derived from typical durations of rescue operations after such events (Appendix D). We skipped earlier t as nearly the half of the evaluated events returned no facts for these t. For the forecast scenario, we started for earthquakes at $t = t_0$+1 d, increasing t by 1 d until day 7. In parallel, we started the forecast range x at 1 d, increasing x by 1 d until 7 d, resulting in $max(t + x)$ =14 d. For floods, we started at $t = t_0$+2 d, increasing t by 2 d until day 14. We started x at 2 d, increasing x by 2 d until 14 d, resulting in $max(t + x)$ =28 d.

For each t evaluated, we aggregate the results $e(t)$ across events by two descriptive measures based on quartiles: the median and the interquartile range (IQR). The median is a robust measure for the central tendency of the results whereas the IQR captures their dispersion [52]. Partitioning the ordered results into four quartiles, the median is the 2nd quartile (Q_2), i.e. the 50th percentile separating the measures into two halves. IQR is the interquartile range, defined by $IQR = Q_3 - Q_1$, the distance between the 25th and the 75th percentile, containing 50 % of the measures.

5.4.4 Results

Real-time Scenario

Figures 5.7 to 5.10 illustrate the real-time quality of our framework, comparing it with the Wikipedia references. They show that our framework is capable of returning current casualty numbers with an average error of less than 20 % on both event types within 7 (30) days elapsed since the earthquake (flood) (Table 5.2). Traces for both types of event indicate no quantitative difference between curve fitting, sliding average, and "last inlier". All three show an average median of around 0.18 for both event types and an IQR of around 0.35 for earthquakes and 0.38 for floods. All other baselines show significant higher average median and IQR values for both event types, especially

Table 5.2: Average error rates for the real-time scenario, covering 7 days elapsed for earthquakes (23 evaluated events) and 30 days elapsed for floods (9 evaluated events).

	Earthquake		Flood	
	⌀Median	⌀IQR	⌀Median	⌀IQR
Fitted curve	0.18	0.37	0.19	0.39
Sliding average	0.18	0.33	0.17	0.35
Last inlier	0.16	0.34	0.17	0.40
Last intra	0.27	0.61	0.24	1.4
Fitted curve (w/ outliers)	0.63	4.4	0.47	3.5
Sliding average (w/ outliers)	0.34	1.5	0.24	1.4

Table 5.3: Spearman's Rho r_s with p-values for earthquakes (33 events, 7 days elapsed for t) and floods (9 events, 30 days elapsed for t). Results per event are based on sliding averages, aggregated across t by median.

	Earthquake		Flood	
Median($e(t)$) versus	r_s	p	r_s	p
Event date (age)	-0.29	0.09	-0.10	0.80
Casualties	-0.02	0.90	0.10	0.80
Event duration	0.39	0.02	0.60	0.09
Search results	-0.17	0.33	0.33	0.38
Documents	-0.39	0.02	-0.03	0.93
Intra-document heterogeneity	-0.03	0.85	-0.23	0.55
Inliers	-0.46	0.01	-0.42	0.26
Outliers	-0.28	0.12	0.37	0.33
Ratio(inliers/outliers)	-0.13	0.47	-0.53	0.14

curve fitting with outliers, having errors twice as much as the others. For the top three strategies, traces for both event types indicate no clear influence of the time elapsed on the achieved error rates. Despite having similar averages for median and IQR, traces for floods display an increased error variance compared to earthquakes.

We also calculated the correlation, i.e. Spearman's Rho r_s [80], between event characteristics and average errors to identify which parameters have a significant influence on the real-time results. We tested the event date (or age), i.e. the time elapsed between the event start and our evaluation, the number of casualties and the event duration, i.e. the time elapsed until the last Wikipedia casualty update. Evaluation parameters per event are the number of search results, i.e. the output of the retrieval module, and the number of documents after applying all framework modules except the fusion module, i.e. the input of the fusion module. The fusion parameters per event are the intra-document heterogeneity, i.e. the average number of relevant relationship tuples per document, the number of inliers, the number of outliers and the inlier/outlier ratio. Table 5.3 shows that none of the tested parameters correlates significantly with the

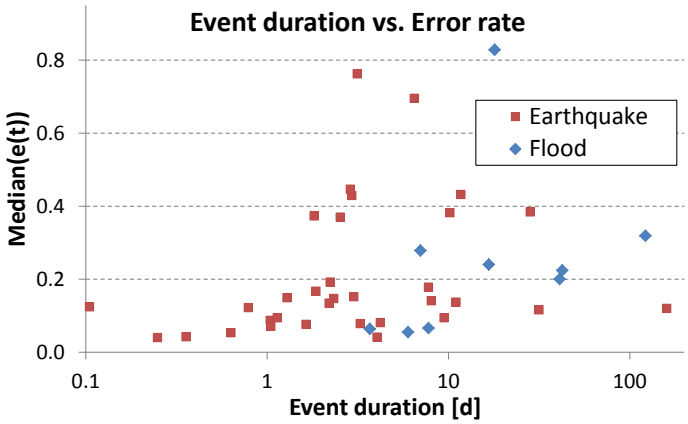


Figure 5.6: Plot of event durations versus error rates for 33 earthquakes (7 days elapsed for t) and 9 floods (30 days elapsed for t). Each data point represents one event. Results per event are based on sliding averages, aggregated across t by median.

Table 5.4: Average error rates for the forecast scenario, covering 14 days elapsed for earthquakes (23 evaluated events) and 28 days elapsed for floods (9 evaluated events).

	Earthquake		Flood	
	⌀Median	⌀IQR	⌀Median	⌀IQR
Fitted curve	0.27	0.58	0.39	0.91
Sliding average	0.29	0.40	0.33	0.45
Last inlier	0.24	0.38	0.39	0.48

error rate. Only results for "event duration", $r_s = 0.39$ and p-value=0.02 for earthquakes, $r_s = 0.60$ and p-value=0.09 for floods, indicate a slight (positive) correlation, i.e. long-lasting events show increased error rates (Figure 5.6).

Forecast Scenario

Given the results for the real-time scenario, we focused our forecasting experiments on the top three strategies: curve fitting, sliding average, and "last inlier". Figures 5.11 to 5.14 illustrate the forecast quality of our framework, comparing it with the Wikipedia references. They show that our framework is capable of returning forecasts with an average error of less than 30 % on earthquakes and less than 40 % on floods within 14 (28) days elapsed since the earthquake (flood) (Table 5.4). Independent of the event type and chosen fusion strategy, we observe two trends when increasing the days elapsed since the event or the number of forecasted days: Increasing the number of days to forecast increases the forecast error and increasing the number of days elapsed decreases the error. Comparing the median error rates for all three fusion strategies on both event types indicates that none outperforms the others consistently. All three show an average median of around 0.27 for earthquakes and around 0.37 for floods. The IQR values imply increased error variances for fitted curves compared to the others: 0.58 versus around 0.39 for earthquakes and 0.91 versus around 0.47 for floods.

5.5 Discussion & Summary

We presented the final stage of our framework, the fusion module. It provides strategies to create reliable, time-resolved views on events despite inconsistencies in the input data. We tested these strategies in two comprehensive case studies tracing the number of casualties reported after 42 natural disasters. To this end, our proposed fusion strategies were embedded into a testbed, namely the entire framework. Consequently, these studies represent an evaluation of the fusion module and the entire framework at the same time. The results showed that our framework is capable of returning current casualty numbers with an average error of less than 20 %. Comparing achieved error rates and event characteristics revealed no significant correlations. As expected, forecasting produced less precise results with average error rates of 30 % to 40 %.

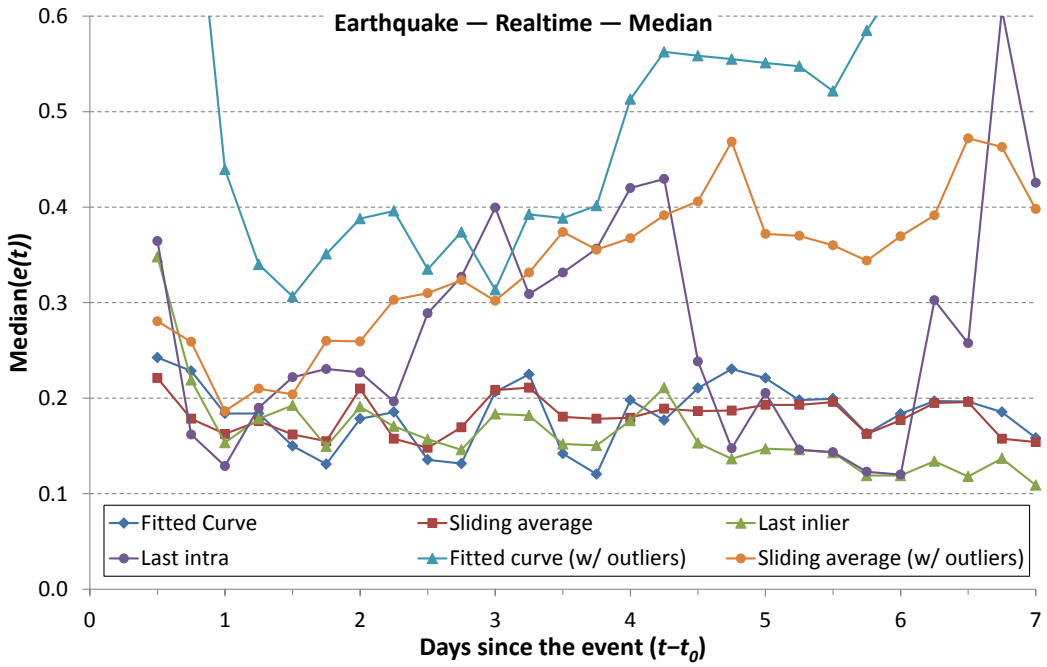
Note that using Wikipedia as reference for time-ware evaluations might disadvantage our framework in terms of measured error rates. The utilized Wikipedia pages are based on information provided by 3rd-party articles, thus the latency of the Wikipedia editing introduces a small bias in the derived time series of casualty numbers. There is always a time gap between the publication of 3rd-party articles and the corresponding manual Wikipedia updates. Consequently, we propagate (biased) Wikipedia update dates as gold standard instead of (correct) publication dates. If our framework retrieves the same news articles or even earlier articles quoting the same value, it will use the (hopefully) correctly identified publication dates, then penalized at the evaluation. This latency-induced bias is hard to quantify in retrospect, as Wikipedia updates often miss any citation of the originating source. For instance, the organizers of the TREC Temporal Summarization Track[8] took these circumstances into account by adding a non-linear "Latency Discount" to their evaluation measure [7].

Overall, these were surprising low error rates given the large number of—probably imperfect—processing steps within the framework and their complex interaction. We think that for many applications these error rates are acceptable, taking the alternative costs of manual information gathering into account. For example, after natural disasters, knowing the scale of casualties is more important for determining the required extend of relief operations than knowing the exact numbers. Our case studies focused on past events due to the availability of reference time series. Archiving similar valuable results at current events requires having current documents on hand, which might be difficult to retrieve by solely using search engine APIs. Recommended framework extensions for targeting current events are further discussed in Section 6.1.

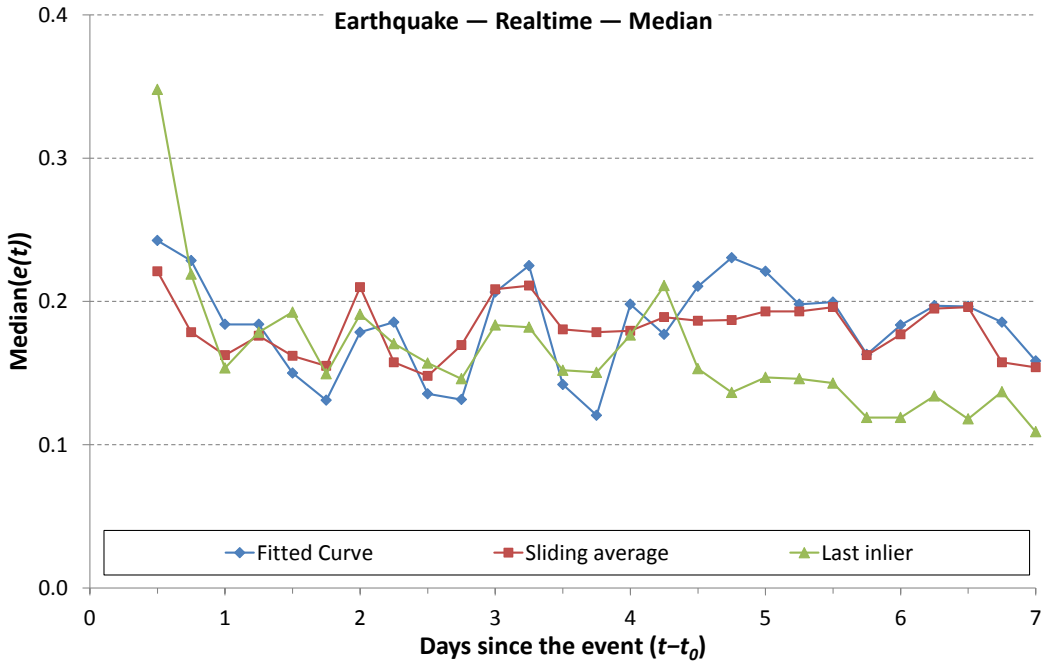
Processing one event by means of our framework[9] took around 140 s for the serial processing of on average 1626 documents. This run-time required that all search results were already retrieved from the search engine API and that the corresponding documents were successfully downloaded and linguistically annotated. This annotation includes (expensive) dependency parsing which takes around 0.3 s per sentence (Table 4.4). Estimating the run-time for "fresh" events is hardly possible, but it can be drastically reduced as the vast majority of the tasks involved in our framework are trivially parallelizable at the document or even at the sentence level.

[8] http://www.trec-ts.org/

[9] All experiments were conducted on a 2012 consumer PC equipped with a 3 GHz CPU.

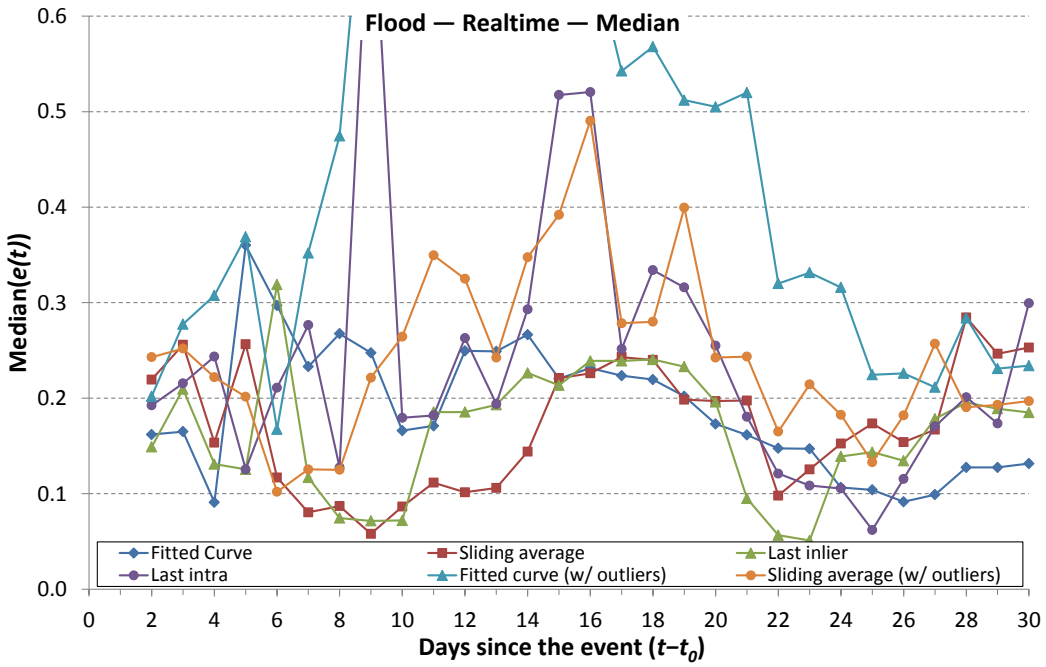


(a) Default fusion strategies, accompanied by all baselines.

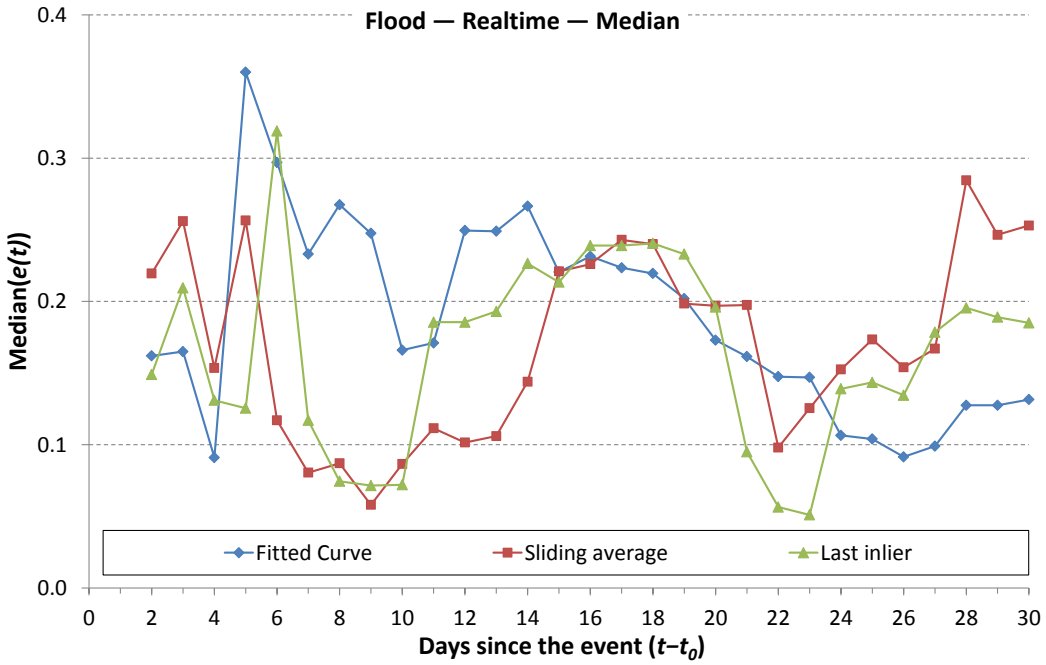


(b) Default fusion strategies, accompanied by "last inlier" (magnified).

Figure 5.7: Real-time differences for earthquakes for fitted curves and sliding averages—the default fusion strategies—compared to the Wikipedia references, accompanied by the baselines. Results are calculated on the evaluation set (23 events), aggregated for each t across events by median.

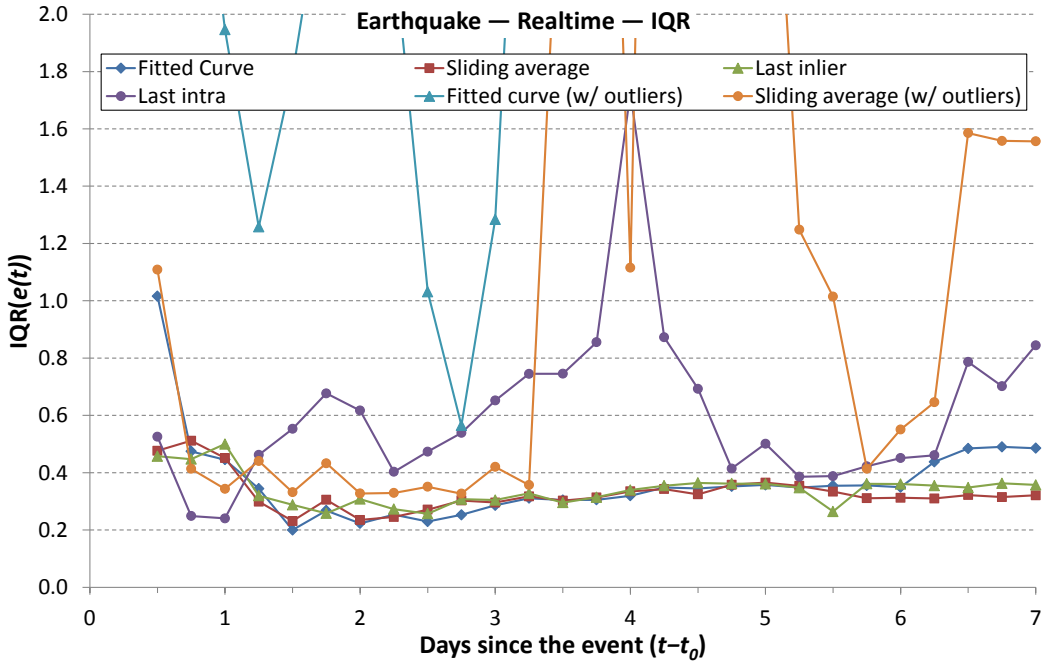


(a) Default fusion strategies, accompanied by all baselines.

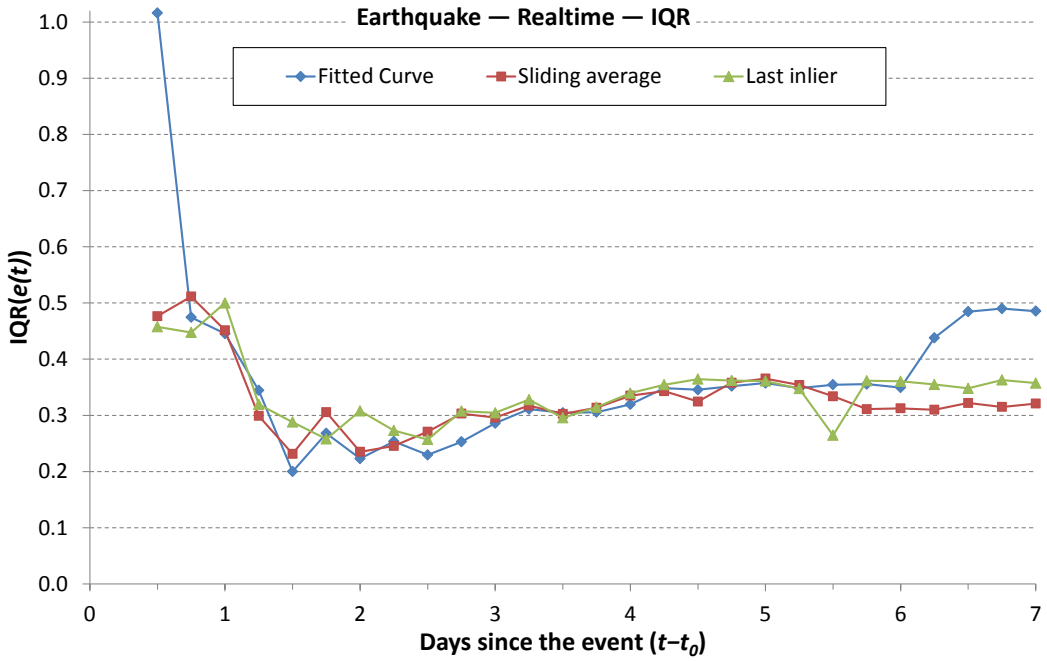


(b) Default fusion strategies, accompanied by "last inlier" (magnified).

Figure 5.8: Real-time differences for floods for fitted curves and sliding averages—the default fusion strategies—compared to the Wikipedia references, accompanied by the baselines. Results are calculated on the evaluation set (9 events), aggregated for each t across events by median.

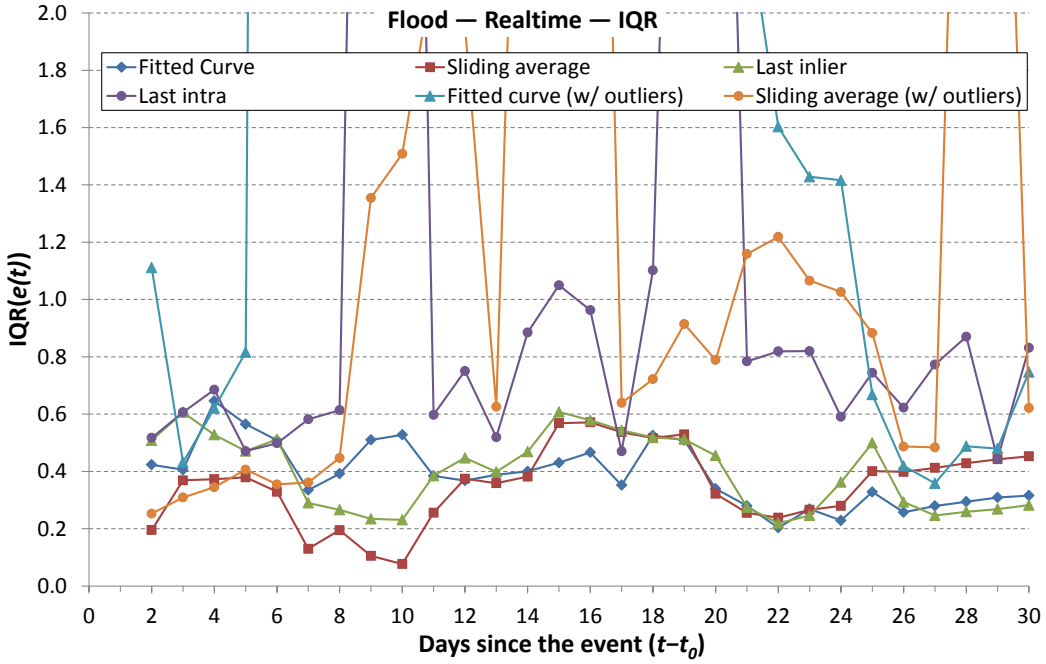


(a) Default fusion strategies, accompanied by all baselines.

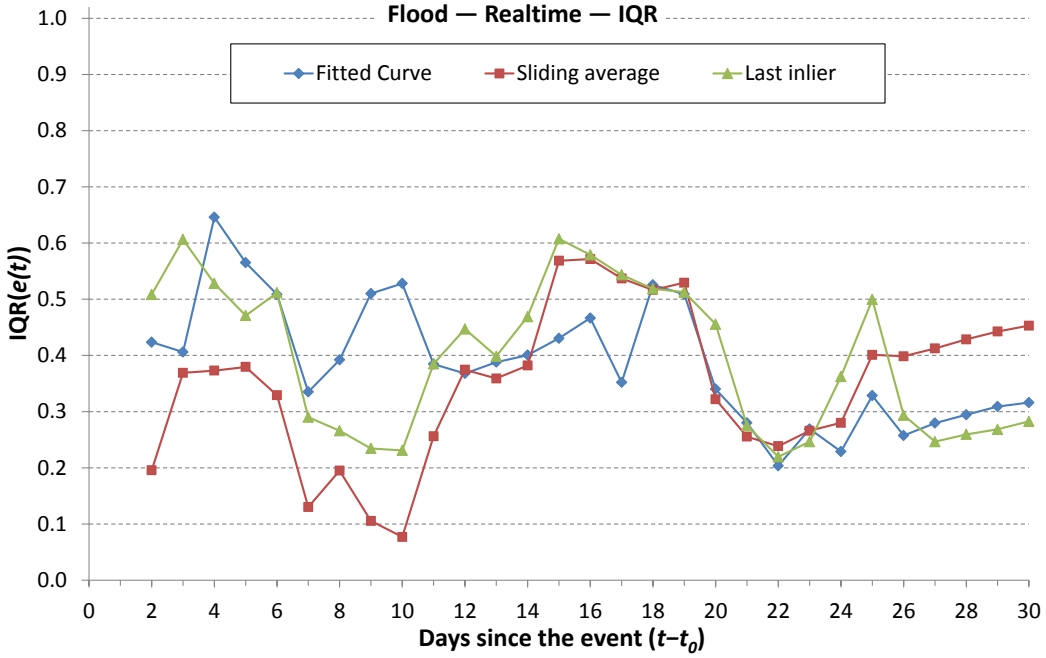


(b) Default fusion strategies, accompanied by "last inlier" (magnified).

Figure 5.9: Real-time differences for earthquakes for fitted curves and sliding averages—the default fusion strategies—compared to the Wikipedia references, accompanied by the baselines. Results are calculated on the evaluation set (23 events), aggregated for each t across events by IQR. IQR is the interquartile range, defined by $IQR = Q_3 - Q_1$, the distance between the 25$^{\text{th}}$ and the 75$^{\text{th}}$ percentile.

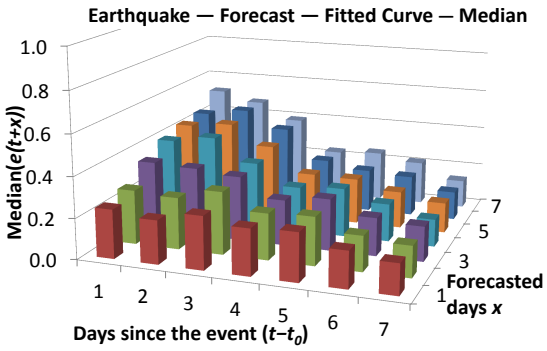


(a) Default fusion strategies, accompanied by all baselines.

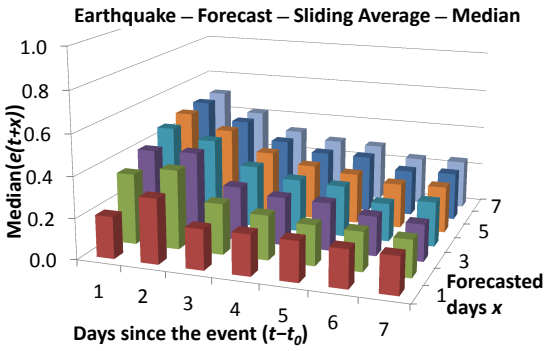


(b) Default fusion strategies, accompanied by "last inlier" (magnified).

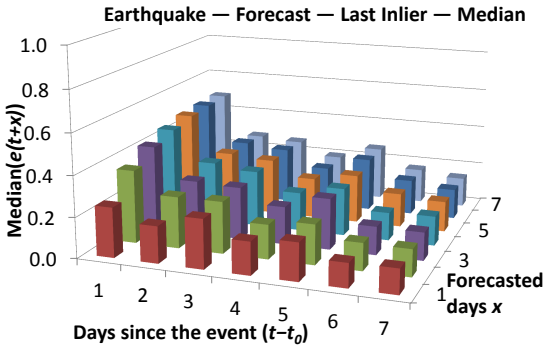
Figure 5.10: Real-time differences for floods for fitted curves and sliding averages—the default fusion strategies—compared to the Wikipedia references, accompanied by the baselines. Results are calculated on the evaluation set (9 events), aggregated for each t across events by IQR. IQR is the interquartile range, defined by $IQR = Q_3 - Q_1$, the distance between the 25^{th} and the 75^{th} percentile.



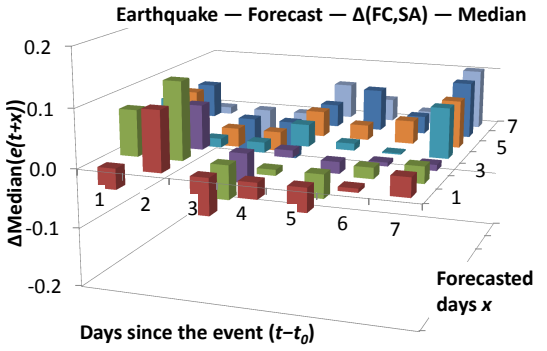
(a) Fitted curve



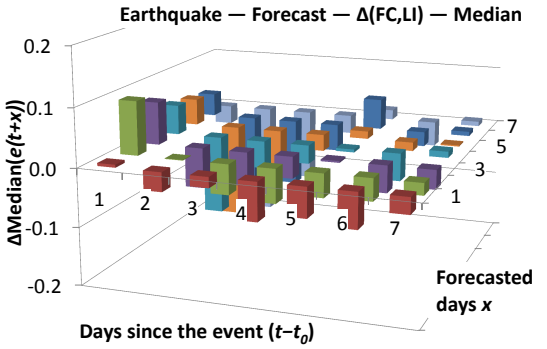
(b) Sliding average



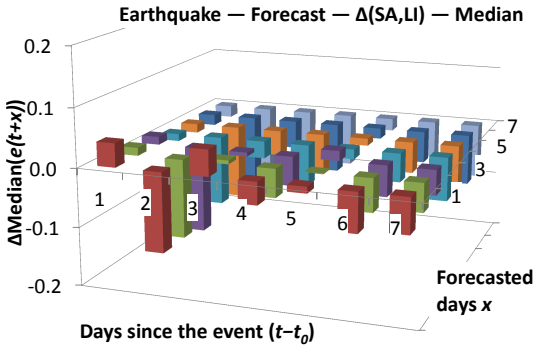
(c) Last inlier



(d) Differences between fitted curves (Fig. 5.11a) and sliding averages (Fig. 5.11b); positive values indicate less forecast errors by fitted curves than sliding averages

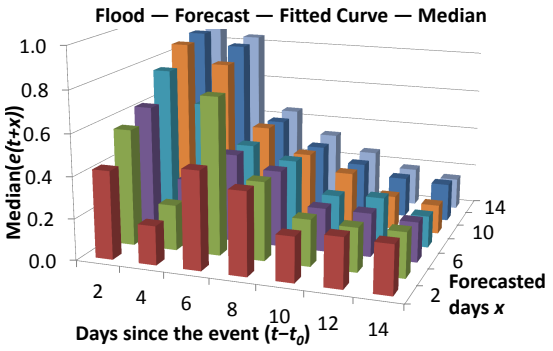


(e) Differences between fitted curves (Fig. 5.11a) and "last inlier" (Fig. 5.11c); positive values indicate less forecast errors by fitted curves than "last inlier"

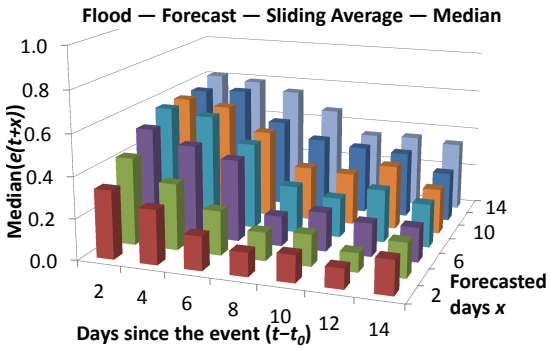


(f) Differences between sliding averages (Fig. 5.11b) and "last inlier" (Fig. 5.11c); positive values indicate less forecast errors by sliding averages than "last inlier"

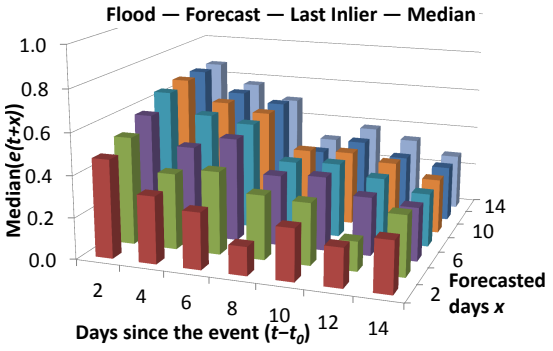
Figure 5.11: Forecast differences for the fusion strategies curve fitting and sliding average compared to the Wikipedia references, accompanied by "last inlier". Results are calculated on the evaluation set (23 events), aggregated for each t across events by median. The value $e(t+x)$ at $(t-t_0, x)$ means: Using all facts from t_0 until t, what is the forecast error at $t+x$? For example, errors for forecasting $x = 5$ days ahead after $t - t_0 = 2$ days elapsed, i.e. at day 7, can be found at (2,5).



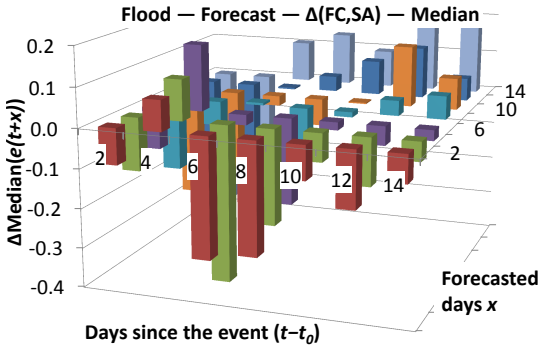
(a) Fitted curve



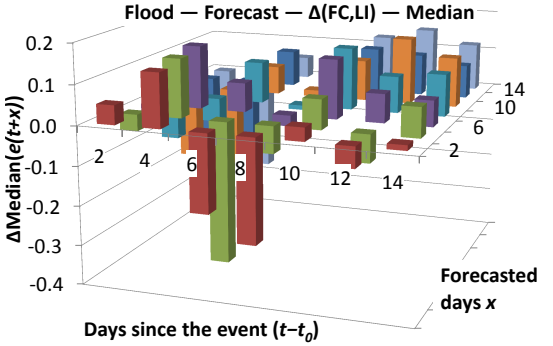
(b) Sliding average



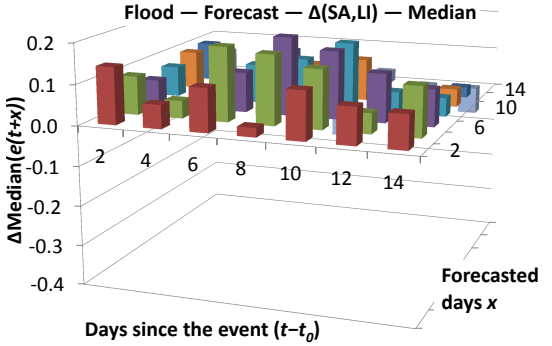
(c) Last inlier



(d) Differences between fitted curves (Fig. 5.12a) and sliding averages (Fig. 5.12b); positive values indicate less forecast errors by fitted curves than sliding averages

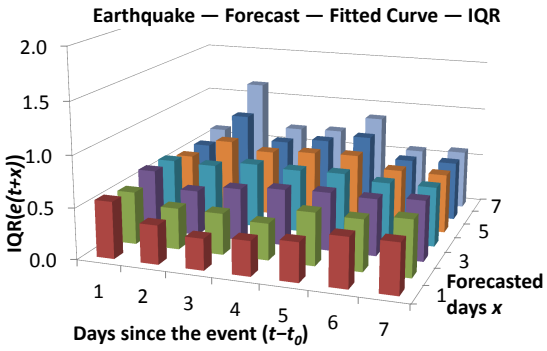


(e) Differences between fitted curves (Fig. 5.12a) and "last inlier" (Fig. 5.12c); positive values indicate less forecast errors by fitted curves than "last inlier"

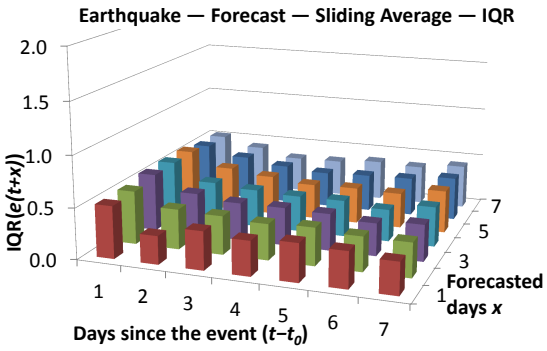


(f) Differences between sliding averages (Fig. 5.12b) and "last inlier" (Fig. 5.12c); positive values indicate less forecast errors by sliding averages than "last inlier"

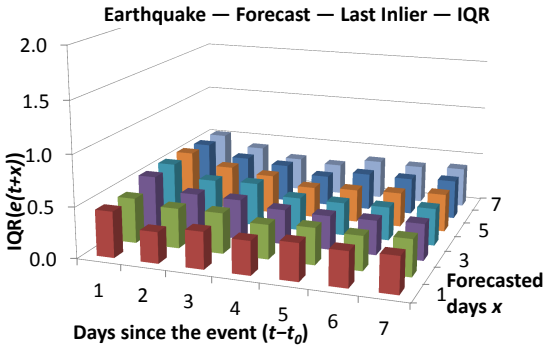
Figure 5.12: Forecast differences for the fusion strategies curve fitting and sliding average compared to the Wikipedia references, accompanied by "last inlier". Results are calculated on the evaluation set (9 events), aggregated for each t across events by median. The value $e(t+x)$ at $(t-t_0, x)$ means: Using all facts from t_0 until t, what is the forecast error at $t+x$? For example, errors for forecasting $x = 6$ days ahead after $t - t_0 = 2$ days elapsed, i.e. at day 8, can be found at (2,6).



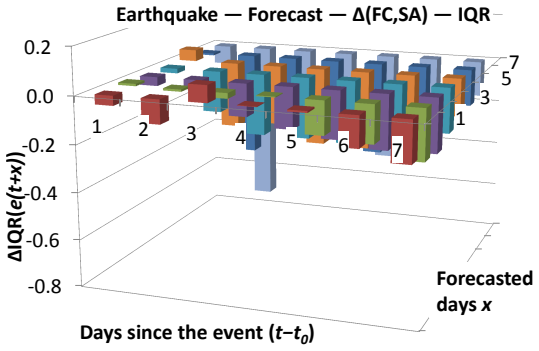
(a) Fitted curve



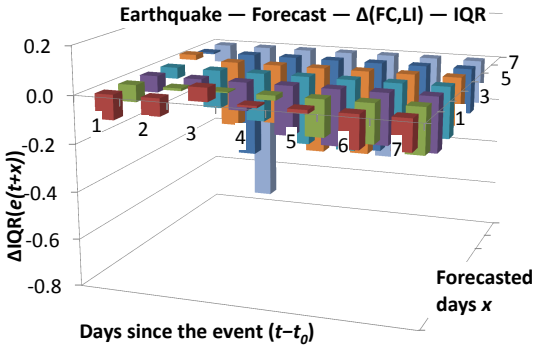
(b) Sliding average



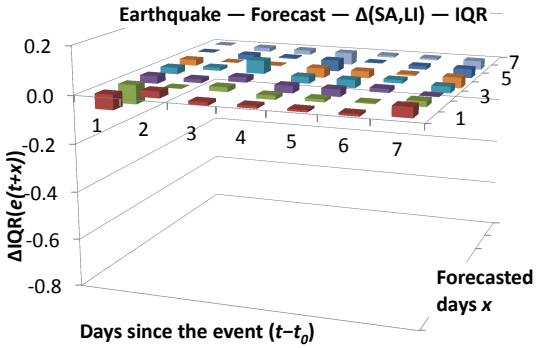
(c) Last inlier



(d) Differences between fitted curves (Fig. 5.13a) and sliding averages (Fig. 5.13b); positive values indicate less forecast errors by fitted curves than sliding averages

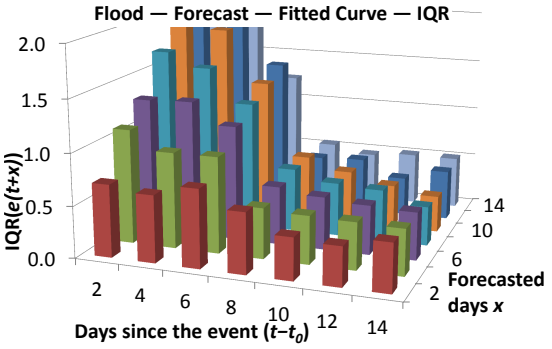


(e) Differences between fitted curves (Fig. 5.13a) and "last inlier" (Fig. 5.13c); positive values indicate less forecast errors by fitted curves than "last inlier"

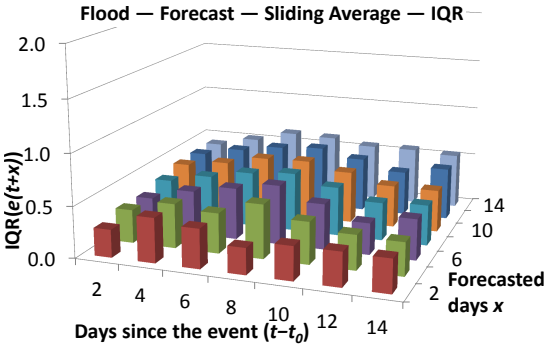


(f) Differences between sliding averages (Fig. 5.13b) and "last inlier" (Fig. 5.13c); positive values indicate less forecast errors by sliding averages than "last inlier"

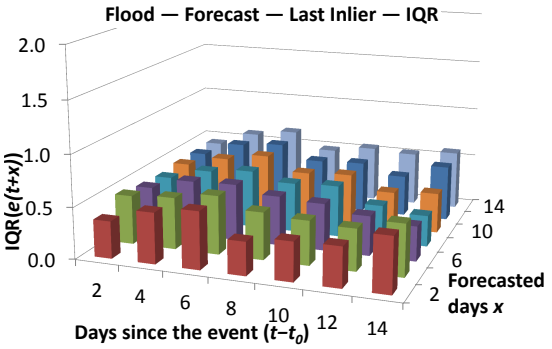
Figure 5.13: Forecast differences for the fusion strategies curve fitting and sliding average compared to the Wikipedia references, accompanied by "last inlier". Results are calculated on the evaluation set (23 events), aggregated for each t across events by the interquartile range IQR. The value $e(t+x)$ at $(t-t_0, x)$ means: Using all facts from t_0 until t, what is the forecast error at $t+x$? For example, errors for forecasting $x=5$ days ahead after $t-t_0=2$ days elapsed, i.e. at day 7, can be found at (2,5).



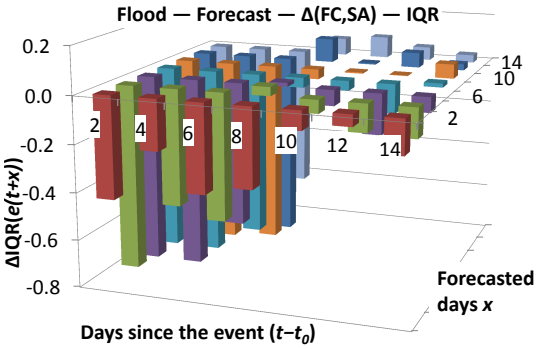
(a) Fitted curve



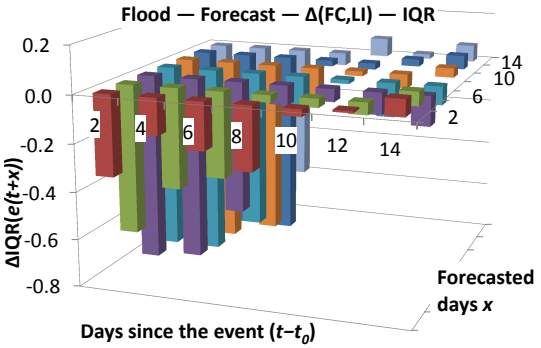
(b) Sliding average



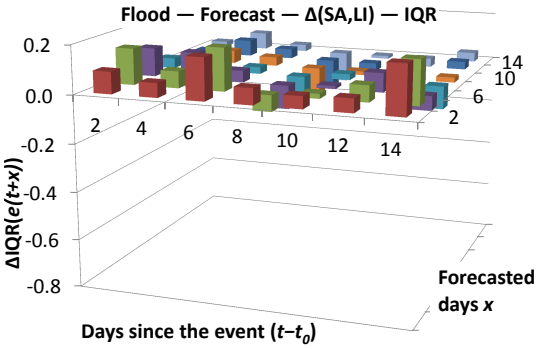
(c) Last inlier



(d) Differences between fitted curves (Fig. 5.14a) and sliding averages (Fig. 5.14b); positive values indicate less forecast errors by fitted curves than sliding averages



(e) Differences between fitted curves (Fig. 5.14a) and "last inlier" (Fig. 5.14c); positive values indicate less forecast errors by fitted curves than "last inlier"



(f) Differences between sliding averages (Fig. 5.14b) and "last inlier" (Fig. 5.14c); positive values indicate less forecast errors by sliding averages than "last inlier"

Figure 5.14: Forecast differences for the fusion strategies curve fitting and sliding average compared to the Wikipedia references, accompanied by "last inlier". Results are calculated on the evaluation set (9 events), aggregated for each t across events by the interquartile range IQR. The value $e(t+x)$ at $(t-t_0, x)$ means: Using all facts from t_0 until t, what is the forecast error at $t+x$? For example, errors for forecasting $x = 6$ days ahead after $t - t_0 = 2$ days elapsed, i.e. at day 8, can be found at (2,6).

Table 5.5: Comparison of the characteristics of the two evaluated event types.

	Earthquake	Flood
Granularity of temporal alignment (t_0)	seconds	day
Regular event duration	≈1 min	weeks
"Final" casualty numbers reported after, i.e. end of rescue operations	5 – 10 d	>1 month
Wikipedia article created after the event (75 % quantile)	within 6 h	>1 week
Number of updates of casualties reported at Wikipedia's infoboxes (avg)	23	11
Granularity of spacial alignment	GPS coordinates	unclear
Affected area	several villages, towns	regions, countries
Worldwide occurrence frequency	rare, no temporal overlaps	common, with temporal overlaps
Relation of *Event occurrence* vs. *Creation of Wikipedia article*	1:1	infrequently

Contrasting the two proposed inter-document fusion strategies—fitted curves and sliding averages—indicated no advantage of the former. Our initial assumption of superior results for fitted curves due encoded typical fact evolutions could not be substantiated. The measured differences for both strategies are marginal, except for forecasting in the flood domain, where sliding averages provided favorable results. Comparing both strategies with the "last inlier" baseline indicated that the outlier detector already returned suitable values for the sampled points in time t. Omitting any inter-document fusion carries the danger of propagating false inliers—outliers classified as inliers—directly to the framework's output, whereas their negative effects are mitigated by the inter-document fusion. For instance, Figure 5.4 shows two false inliers near day 12, resulting in "last inlier" returning the false value 1144 for 65 h, resulting in 50 % error rate. The fitted curve and sliding average are less affected, resulting in an error rate of less than 15 %. Results for the other baselines, "last intra" and "with outliers", emphasized the need for outlier detection, especially for curve fitting due to its inherent sensitivity to outliers. Consequently, we recommend applying both: outlier removal and inter-document fusion.

Contrasting the two event types, we observed increased error rates and/or error variances for flood events. Table 5.5 compares the type's key characteristics. We identified the following reasons most likely causing the observed differences:

Precision of t_0 While beginnings of earthquakes are reported to the split second, beginnings of floods are difficult to define due to the cumulative nature of water levels. Using estimated t_0 values for floods added a significant bias to our time-sensitive evaluations, not present for earthquakes (Section 5.4.1).

Suitability of the framework configuration Lacking training data, we had to reuse the earthquake fusion configuration for floods. Using an optimized configuration would probably had improved the results.

Size of evaluation set Testing only nine flood events may lead to increased error variances. Floods are the most frequent natural disasters, but their "usualness" is probably causing little attraction to Wikipedia authors.

Temporal overlap between events Floods are the most frequent natural disasters, occurring regularly, and usually last multiple weeks. Earthquakes are "rare" in comparison and reports may be published for two weeks only. Consequently, there is a chance of temporal overlaps between floods not present for earthquakes, potentially interfering at the retrieval of event-relevant documents.

5.6 Related Work

Closely related to our work on time-aware information fusion is temporal summarization, spurred by the TREC Temporal Summarization Tracks[10] held in 2013–2015. The task's goal was "to develop systems which can detect useful, new, and timely sentence-length updates about a developing event" [7]. Given such an event—defined by type (e.g. accident, earthquake, protest), start/end date, and query terms—and a large corpus of timestamped documents, systems were required to generate a series of such updates describing the evolution of the event over time. Reference sentences, so-called "information nuggets" were manually derived from Wikipedia revisions by the track organizers. Example nuggets for the 2012 Hurricane Sandy[11] are 'hurricane force wind warnings are in effect from Rhode Island Sound to Chincoteague Bay' or 'over 5000 commercial airline flights scheduled for October 28 and October 29 were cancelled'. The timestamps of the corpus documents are defined by the download dates. Temporal summarization connects information retrieval with document summarization and time as key dimension. As a general pattern, participating systems performed the following processing steps: (1) filter incoming documents for relevance based on the event (retrieval), (2) extract salient sentences (summarization), (3) compare new sentences with old sentences (de-duplication), and (4) emit new sentences as updates. All these steps needed to be performed in an online-setup, accessing the corpus as stream of documents ordered in time and emitting new updates as quickly as possible. The organizers defined custom evaluation metrics based on relevance, coverage, novelty, and latency of the updates. The best performing system in 2014 scored 0.1531 using a mean measure analogous to F1 [7, 65]. In contrast to our work, they eliminate duplicates whereas duplicates play an important role in our fusion strategies by acting as majority voters.

Steinberger et al. created the Europe Media Monitor[12], providing an explorative access to more than thousand news sources in multiple languages [120]. These sources are crawled regularly, the download date may serve as document timestamp if no meta data, such as RSS feeds, are available. Their system eases manual information browsing

[10] http://www.trec-ts.org/
[11] https://en.wikipedia.org/wiki/Hurricane_Sandy
[12] http://emm.newsbrief.eu/overview.html

by automatic recognition of persons, organizations, etc. as well as categorizing and clustering the documents. They also detect trending topics among their monitored media, tracing their development over time and in space. Still, topic-relevant documents need to be analyzed manually to gather contained key facts. Compared to their work, we "zoom in" instead of "zoom out": moving from documents to facts instead of to topics, shifting from media monitoring to "fact monitoring".

Assessing damages after natural disaster is important to coordinate relief operations. Remote sensing techniques, such as optical, LiDAR[13], or SAR[14], are of great help here [34]. For instance, high-resolution satellite images offer a broad as well as a detailed view on infrastructural damages without field surveys. Acquired time series data enables comparison of the effected area before and after the event, e.g. for earthquakes [112] or floods [25, 118]. Such image data allows to identify collapsed buildings or destroyed bridges. The former indicates where to search for trapped people and the latter is important for reaching identified areas. Another approach to assess the required scale of relief operations is to estimate the number of casualties based on the physical parameters of the event and a geophysical model of the area. For example, WAPMERR [15] provides loss estimates for earthquakes and publishes prompt alerts within 30 minutes after the event. They utilize a dataset about the worldwide population distribution and conditions of buildings. Based on this data and the location of the earthquake, its depth and magnitude, they are able to recognize over 70 % of the majors disasters [136]. Using static thus potentially outdated population data may lead to false estimates for areas with large changes in the number of inhabitants, e.g. growing cities due to rural depopulation. Generating estimates based on current media reports as in our work does not rely on potentially outdated data, but requires far more than 30 minutes elapsed.

[13] *Li*ght *D*etection *A*nd *R*anging: uses pulsed lasers to measure distances to objects
[14] *S*ynthetic *A*perture *R*adar: imaging radar, able to acquire data through clouds
[15] `http://www.wapmerr.org/realtime.asp`

6 Summary & Outlook

Finding reliable information about given events from large and dynamic text collections, such as the web, is a topic of great interest. For instance, rescue teams and insurance companies are interested in concise facts about damages after disasters, which can be found today in web blogs, online newspaper articles, social media, etc. Knowing these facts helps to determine the required scale of relief operations and supports their coordination. However, the extraction and aggregation of event data from the web is a complex task. It includes (1) source identification, (2) temporal document classification, (3) fact extraction, and (4) time-aware aggregation of these facts.

In this thesis, we have presented a configurable framework for automatically extracting and aggregating event data from the web. Our work aims at easing the process of gaining situational awareness during exceptional and long-lasting events, such as natural disasters. It focuses especially on the temporal aspect of facts describing such events as they change over time, potentially leading to inconsistencies. We presented and evaluated techniques and solutions for each of the outlined challenges, embedded in a four-step framework. Applied methods were based on pattern matching, natural language processing, and machine learning. Retrieving relevant documents from the web gave precision results up to 78 %, aligning documents in time resulted in up to 91 % accuracy and extracting casualty reports from news articles achieved up to 78 % F1-measure. We evaluated our framework in two case studies in the disaster domain, tracing the evolution of casualty numbers reported on the web. The results show that our framework is capable of automating this task. Gathering current numbers resulted in average differences below 20 %, sufficient for many applications, e.g. assessing the required scale of relief operation. Even forecasting future numbers is feasible, resulting in differences of 20 % to 60 %.

6.1 Future Research Directions

The achieved accuracies demonstrate the tractability of the general problem of extracting and aggregating event data from the web, motivating further research on this topic. Taking the current framework as a starting point, we see two main future research directions: improvements and extensions. Section 6.1.1 focuses on improving the current components whereas Section 6.1.2 proposes enhancements to increase the applicability of the framework.

6.1.1 Improvements

To increase the number of relevant documents retrieved, generated queries should include terms referring to the information request instead of only event-relevant terms, e.g. 'wounded' or 'injury' to find reports covering injured people.

Regarding temporal alignment, determining correct time zones and disambiguating date formats are the most important starting points to improve PcDE. Given the URL, the time zone might be derived from the top-level domain, the sitemap[1], or the RSS feed[2] of the site. The latter two may contain the publication date, and if so, in a well defined format (ISO8601). Hints to parse ambiguous dates correctly might be acquired by inspecting other pages originating from the same web site containing unambiguous dates. Another promising source are external information, such as URL shorteners, providing upper or lower bounds for the publication date [114]. These limits might help to narrow down parsing possibilities.

Our evaluation of relationship extraction indicated, that results are limited by the preceding named entity recognition. An alternative to just improve entity recognition are jointed extraction approaches [63]. By recognizing entities and relationship tuples simultaneously, they carry the potential to overcome one deficit of extraction pipelines: its unidirectional NER→RE information flow. Entities and tuples dependent on each other, so recognizing them in one step is reasonable.

Improved information fusion results might be achievable by assessing the trustworthiness of incoming documents. Currently, we have two indicators for validity available per document: the search rank and the originating site. Both could be applied during outlier detection and inter-document fusion.

6.1.2 Enhancements

Applying our framework to other information needs, for example to extract infrastructural damages reported on social media during floods, requires substantial enhancements.

Self-Crawling

Relying solely on search engine results when applying our framework to current, ongoing events might be insufficient in terms of retrieving current information. Consequently, the retrieval module should be capably of self-crawling, for instance limited to specific sites [120].

Spatiotemporal Visualization

The current output of the framework are, more or less, numbers. Instead, processed information should be visualized based on the temporal (and spacial) context of the information. For instance, pinning reports on infrastructural damages on to maps along a time line [39, 98].

User-generated Content

User-generated contents were successfully applied to gain situational awareness during crisis events [57]. Utilizing humans as “social sensors” may lead to timely, actionable information [4]. Challenges here are the retrieval of event-relevant messages [113] and

[1] `https://www.sitemaps.org/index.html`
[2] `http://www.rssboard.org/rss-specification`

the processing of informal language [43, 101]. Furthermore, relying on user-generated content carries the danger of propagating false claims ("fake news") to framework users [28, 131].

Non-English Documents

Using non-English content as additional source would enable accessing informative reports from (local) media, published in the language of the country. Supporting other languages beside English would require little effort for most modules, except for fact extraction. Applied extraction methods here depend on language-specific natural language processing routines, such as POS tagging or dependency parsing, and/or on annotated training data for the targeted facts.

Open Information Extraction

Following the idea of generalizing our framework, it would be very interesting to include domain-independent fact extraction methods, moving towards Open Information Extraction [9, 38, 110]. The most challenging aspect is the succeeding fusion of the extracted (untyped) facts: identifying semantically compatible facts and developing adequate fusion strategies.

A PcDE's Date Patterns

These are the patterns utilized in PcDE to parse date expressions (Section 3.1), ordered by precedence. Each day pattern (Section A.1) is combined with each time pattern (Section A.2) in both possible orders, optionally followed by a time zone notation: "day time" or "time day" as well as "day time zone", "time day zone", or "time zone day". In addition, the two ISO 8601[1] formats yyyy-MM-dd'T'HH:mm:ss'Z' and yyyy-MM-dd'T'HH:mm'Z' are added to the pattern catalog, resulting in 1052 patterns. Literals used refer to Java's SimpleDateFormat[2] syntax.

Literal	Meaning	Value range
`y`	Year	95-17, 1995-2017
`M`	Month in year	1-12, 01-12
`d`	Day in month	1-31, 01-31
`a`	Am/pm marker	'am', 'pm'
`H`	Hour in day	0-23
`h`	Hour in am/pm	1-12
`m`	Minute in hour	00-59
`s`	Second in minute	00-59

A.1 Day Patterns

If the UK format is selected for parsing, the precedence is: (1) UK-specifc, (2) US-specific, (3) general; for the US it is: (1) US-specifc, (2) UK-specific, (3) general.

UK-specific Day Patterns

`d/M/yy, d/M/yyyy, dd-MM-yy, dd-MM-yyyy, dd.MM.yy, dd.MM.yyyy`

US-specific Day Patterns

`M/d/yy, M/d/yyyy, MM-dd-yy, MM-dd-yyyy, MM.dd.yy, MM.dd.yyyy`

General Patterns

`d MMM yyyy, MMM d yyyy, yyyy-MM-dd, yyyy/MM/dd, yyyy.MM.dd,`
`yyyy MMM d, yy-MM-dd, yy.MM.dd, yy/MM/dd`

A.2 Time Patterns

`h:mm:ss aa, h:mm:ssaa, HH:mm:ss, h:mm aa, h:mmaa, h.mm aa, h.mmaa,`
`HH:mm, HH.mm, HHmm`

[1] https://en.wikipedia.org/wiki/ISO_8601
[2] https://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html

B Annotation Guidelines

Es soll untersucht werden, in wieweit es möglich ist, Angaben zu Schäden an Menschen in englischen Textmeldungen über Erdbeben automatisch zu erkennen. Dazu vorab ein Beispiel (im Annotationsformat mit Notepad++-Syntaxhervorhebung, s.u.):

```
Both_<>_<> provinces_<>_<> were_<>_<> severely_<>_<> affected_<>_<> by_<>_<> a_<>_<>
devastating_<>_<> earthquake_<>_<> in_<>_<> May_<>_<> which_<>_<> left_<>_<> almost_<M>_<R2>
70,000_<Qk>_<R2> people_<O>_<R2> dead_<St>_<R2> ._<>_<>
```

In diesem Satz sollen die Entitäten `almost`, `70,000`, `people`, `dead`, sowie die Beziehung `R2` zwischen ihnen gefunden werden.

B.1 Korpus

Jedes Dokument befindet sich in einer eigenen Datei. Als Kodierung wird UTF-8 verwendet. Die Bearbeitung muss ausschließlich mit einem UTF-8-fähigen Editor erfolgen. Sollte beim Öffnen die Datei mit den Zeichen `ï"¿` beginnen, so bietet das verwendete Programm nur eine unzureichende UTF-8-Unterstützung.

Im Prinzip ist bei modernen Betriebsystemen die Verwendung des bordeigenen Texteditors ausreichend. Um eine Annotationsvervollständigung und Syntaxhervorhebung nutzen zu können, wird jedoch die Verwendung des mitgelieferten und speziell angepassten Editors Notepad++[1] unter Windows empfohlen. Alle Annotationsbeispiele geben die Ansicht in diesem Editor wieder.

B.1.1 Dokumentaufbau

Auszug aus einem Korpusdokument:

```
#uri http://news.bbc.co.uk/2/hi/asia-pacific/7591152.stm
#encoding UTF-8
#received Mon Mar 01 17:02:44 CET 2010

## ein Dokumentkommentar

## ein Satzkommentar
China_<>_<> earthquake_<>_<> death_<>_<> toll_<>_<> rises_<>_<>

...
```

[1] https://notepad-plus-plus.org/

Im Kopf befinden sich Metaangaben (alle Zeilen beginnend mit #). Diese bitte unverändert lassen. Danach folgt der eigentliche Text, ein Satz pro Zeile mit einer Leerzeile.

Jeder Satz ist aufgeteilt in Token, welches es zu annotieren gilt. Dazu befinden sich am Ende jedes Tokens zwei Klammerpaare _<>_<>. Das erste Paar nimmt die Entitätsannotation auf, das zweite die Relationsannotation. Wie das zu erfolgen hat, folgt im nächsten Abschnitt.

Zusätzlich können, beginnend mit ##, Kommentare in das Dokument eingefügt werden. Stehen diese direkt oberhalb eines Satzes, so gelten sie als zum Satz gehörend, sonst zum Dokument.

B.2 Annotation

In der Arbeit wird auf fünf Entitätstypen

OBJEKT	Wer ist betroffen?
QUANTITÄT & MODIFIKATOR	Wie viele sind betroffen?
SCHADENSINDIKATOR	Welcher Schaden/Auswirkung trat auf?
NEGATION	Enthält die Schadensangabe eine Negation?

und eine Relation

```
ERDBEBENSCHADEN (
  NEGATION, MODIFIKATOR, QUANTITÄT, OBJEKT, SCHADENSINDIKATOR
)
```

über all diese fokussiert.

Nachfolgend das zu verwendende Tagset mit Hinweisen.

B.2.1 Entitäten

Entitäten sind eine Sequenz benachbarter Token eines Satzes (kein Lücken, nicht überlappend, nicht satzübergreifend). Damit gehört auch jedes Token höchstens einer Entität an.

Jede Entität ist an mindestens einer Relationsinstanz – auch Relationstupel genannt – beteiligt, mehrere sind möglich (siehe Beispiel unter Hinweis #2). Es kann also kein Token als Entität annotiert werden ohne dieses gleichzeitig einem Relationstupel zuzuordnen.

B.2.1.1 Objekt

Dient der Markierung der geschädigten Person(en). Hier drückt sich auch die Beschränkung der Arbeit auf Menschen aus.

Tag:	`O` (wie Objekt)
Beispiele:	`ambulances, canadians, children, engineer, families, members, mothers, people, producer, residents, seamstresses, students, vendors, workers, lives`

```
## 'people' ist ein häufig auftretendes Objekt, siehe auch B.2.2.2
One_<>_<> of_<>_<> the_<>_<> largest_<>_<> earthquakes_<>_<> on_<>_<> record_<>_<>
killed_<St>_<R3> more_<M>_<R3> than_<*>_<> 700_<Qk>_<R3> people_<O>_<R3> in_<>_<>
Chile_<>_<> on_<>_<> Saturday_<>_<> ._<>_<>
```

Das Wort `others` ist häufig relativ (siehe Abschnitt 3.1.2.3) und gehört mehr zur Quantitätsangabe, als das Objekt zu beschreiben und ist entsprechend nicht als `O` zu annotieren.

Bei Multi-Token-Ausdrücken ist der Oberbegriff zu verwenden (siehe auch Hinweis #6).

```
## 'members' statt 'members of the Hoy Mismo staff'
## 'producer' statt 'producer Ernesto Villanueva'
## 'engineer' statt 'engineer David Mendoza Corcega'
Some_<Qv>_<R33> members_<O>_<R33> of_<>_<> the_<>_<> Hoy_<>_<> Mismo_<>_<> staff_<>_<>
died_<St>_<R33,R34,R35> ,_<>_<> including_<>_<> producer_<O>_<R34> Ernesto_<>_<>
Villanueva_<>_<> and_<>_<> engineer_<O>_<R35> David_<>_<> Mendoza_<>_<> Corcega_<>_<> ,_<>_<>
who_<>_<> had_<>_<> just_<>_<> parked_<>_<> at_<>_<> the_<>_<> Televisa_<>_<> building_<>_<> ,_<>_<>
but_<>_<> had_<>_<> no_<>_<> time_<>_<> to_<>_<> escape_<>_<> from_<>_<> his_<>_<> car_<>_<> ._<>_<>
```

Wenn möglich und sinnvoll, sollen die Elemente einer Objektaufzählung[2] als Multi-Token-Entität annotiert werden.

```
## 'students and teachers' bilden eine Einheit; sie beziehen sich auf den selben
Schaden und werden als Multi-Token-Entität annotiert
More_<M>_<R4> than_<*>_<> 1,000_<Qk>_<R4> off_<>_<> the_<>_<> Middle_<>_<> School_<>_<>
's_<>_<> students_<O>_<R4> and_<*>_<> teachers_<*>_<> died_<St>_<R4> in_<>_<> the_<>_<>
earthquake_<>_<> ._<>_<>
```

B.2.1.2 Quantität

Beschreibt die Anzahl der Geschädigten. Diese Entität wird weiter unterteilt.

[2] Dies kann eine ODER- oder UND-verknüpfte Aufzählung sein.

B.2.1.2.1 Kardinal

Ist für alle Ganzzahlen zu verwenden.

Tag: `Qk` (wie Quantität kardinal)
Beispiele: `12, 1,879, ten, a, 1.3 million, no` (im Sinne von *keine*), `another`

```
## '9,600' und '1,879' sind kardinale Quantitäten
9,600_<Qk>_<R17> injured_<Sl>_<R17,R18> people_<O>_<R17,R18> received_<>_<>
treatment_<>_<> ,_<>_<> including_<>_<> 1,879_<Qk>_<R18> who_<>_<> needed_<>_<>
hospitalization_<>_<> ._<>_<>
```

Neben allen Kardinalzahlen sind damit auch solche in Dezimalschreibweise wie `1.3 million` zu erfassen.

Auch wenn `a` und `another` als `Qk` für one verstanden werden, sind andere Wörter, die indirekt anzeigen, dass es sich um genau eine Person handelt (`the`, Possessivpronomen, ...), nicht als `Qk` zu annotieren.

B.2.1.2.2 Ordinal

Ist für alle Ordnungszahlen zu verwenden.

Tag: `Qo` (wie Quantität ordinal)
Beispiele: `second`, `10th`

```
## 'third' ist eine ordinale Quantität
This_<>_<> was_<>_<> the_<>_<> third_<Qo>_<R7> victim_<O>_<R7> found_<>_<>
dead_<St>_<R7>._<>_<>
```

B.2.1.2.3 Vage

Damit sind alle ungenauen Mengenangaben zu annotieren.

Tag: `Qv` (wie Quantität vage)
Beispiele: `dozens, high, hundreds, low, many, some`

```
## 'many' ist eine vage Quantität
Within_<>_<> minutes_<>_<> ,_<>_<> the_<>_<> steel-frame_<>_<> structure_<>_<> collapsed_<>_<>
,_<>_<> crushing_<St>_<R19> and_<>_<> trapping_<Ss>_<R20> many_<Qv>_<R19,R20>
people_<O>_<R19,R20> inside_<>_<> ._<>_<>
```

Hinweise auf fehlende Informationen über Quantitätien sind keine `Qv`s und werden entsprechend nicht annotiert:

```
## 'unclear how many' ist keine Quantitaetsangabe!
It_<>_<> was_<>_<> unclear_<>_<> how_<>_<> many_<>_<> died_<>_<> ._<>_<>
```

Ebenfalls keine `Qv`s sind prozentuale (z. B. `60% of the population`) oder relative Angaben (z. B. `fewer injured`), auch nicht relativ zu vorher genannten Quantitäten. Diese sind im Rahmen der Annotation zu ignorieren.

```
## 'all others died' nicht annotiert, weil relativ
Five_<Qk>_<R5> people_<O>_<R5> were_<>_<> killed_<St>_<R5> in_<>_<> a_<>_<> landslide_<>_<>
,_<>_<> but_<>_<> virtually_<>_<> all_<>_<> others_<>_<> died_<>_<> in_<>_<> flash_<>_<>
flooding_<>_<> ._<>_<>
```

Eine Ausnahme bilden Konstruktionen, bei denen einige der Tokens so annotiert werden können, dass keine relative, sondern eine aus dem Kontext erkennbar wahre, absolute Aussage übrig bleibt:

```
## Many wird hier nicht relativ, sondern absolut verstanden
Many <Qv> <R11> of <> <> the <> <> dead <St> <R11> are <> <> children <O> <R11> . <> <>
```

In diesem Beispiel ist aus dem Satz vorher klar, dass es eine sehr große Anzahl Toter gab. Da es bei einer großen Teilmenge dieser Vielen immer noch gerechtfertigt ist von einer großen Menge zu sprechen, kann das relative `of` ignoriert werden und im Satz die Instanz `many dead children` annotiert werden.

B.2.1.2.4 Andere

Dieses Tag ist ein Sammelbecken für alles, was nicht in die anderen Kategorien passt. Bei Verwendung bitte den Satz mit einem Kommentar versehen.

Tag: `Qa` (wie Quantität andere)

Relative Mengenangaben sind keine `Qa`s, in Ausnahmefällen können Teile von ihnen allerdings als `Qv`s annotiert werden (siehe Beispiel in Abschnitt B.2.1.2.3).

B.2.1.2.5 Bereichsangabe

Tag: `Q(k|o|v|a)B` (wie Quantität ... Bereichsangabe)

Regulär sind Bereichsangaben wie `between 66,000 and 242,000` wie andere in Beziehung stehende Relationstupel als zwei getrennte Tupel zu annotieren.

```
## '7,000' und '35,000' sind eine Bereichsangabe und werden mit zwei
Relationsinstanzen erfasst
When <> <> the <> <> government <> <> did <> <> give <> <> estimates <> <> of <> <> the <> <>
number <St> <R12,R12X> dead <*> <> , <> <> they <> <> vacillated <> <> between <> <>
7,000 <Qk> <R12> to <> <> 35,000 <Qk> <R12X> people <O> <R12,R12X> . <> <>
```

Die Verwendung des Appendix `B` ermöglicht eine Reduzierung des Annotationsaufwands. Folgendes ist äquivalent[3]:

```
## äquivalent mit einer Relationsinstanz und dem Appendix B
When <> <> the <> <> government <> <> did <> <> give <> <> estimates <> <> of <> <>
the <> <> number <St> <R12> dead <*> <> , <> <> they <> <> vacillated <> <> between <> <>
7,000 <QkB> <R12> to <> <> 35,000 <QkB> <R12> people <O> <R12> . <> <>
```

Mit der ersteren, expliziten Annotationsform werden Bereichsangaben annotiert, bei denen für die QUANTITÄTen abweichende MODIFIKATORen gelten.

```
## '50,000' und '200,000' sind eine Bereichsangabe mit abweichenden Modifikatoren
und werden daher als zwei Relationsinstanzen erfasst, nicht mit QkB
Estimates <> <> of <> <> the <> <> number <St> <R0,R1> killed <*> <> in <> <> last <> <>
Tuesday <> <> 's <> <> earthquake <> <> vary <> <> from <> <> 50,000 <Qk> <R0> to <> <>
at <M> <R1> least <*> <> 200,000 <Qk> <R1> . <> <>
```

[3] Anmerkung des Autors: Diese Form wird intern wieder in die explizite Form mit zwei Relationsinstanzen überführt.

B.2.1.3 Modifikator

Diese Entität setzt sich aus den Token zusammen, welche den Wert einer QUANTITÄT verändern. Das sind in der Regel Adverben, können aber auch Verben sein.

Tag:	`M` (wie Modifikator)
Beispiele:	`about, almost, approximately, around, at least, more than, over tops, passes`

Ein MODIFIKATOR bezieht sich immer auf eine QUANTITÄT.

```
## 'almost' ist ein Modifikator und bezieht sich auf '70,000'
Both <>_<> provinces <>_<> were <>_<> severely <>_<> affected <>_<> by <>_<> a <>_<>
devastating <>_<> earthquake <>_<> in <>_<> May <>_<> which <>_<> left <>_<> almost <M> <R2>
70,000 <Qk> <R2> people <O> <R2> dead <St> <R2> . <>_<>
```

Eine Wertung (z. B. `only`) ist kein MODIFIKATOR.

Bei Bereichsangaben: Formulierungen wie `between`, welche eine Bereichsangabe (siehe B.2.1.2.5) nur begleiten aber die QUANTITÄT nicht relativieren sind keine MODIFIKATORen.

```
## 'vacillated between' ist kein Modifikator
When <>_<> the <>_<> government <>_<> did <>_<> give <>_<> estimates <>_<> of <>_<>
the <>_<> number <St> <R12> dead <*> <>_<> , <>_<> they <>_<> vacillated <>_<> between <>_<>
7,000 <QkB> <R12> to <>_<> 35,000 <QkB> <R12> people <O> <R12> . <>_<>
```

Es liegt in der Natur des Problems, dass oft über das Ausmaß der Schäden nur gemutmaßt werden kann. Solche Schätzungen sind normal zu annotieren. Ein Wort, welches dabei die Schätzung ausdrückt (z. B. `may`, `estimated`) ist kein MODIFIKATOR.

```
## 'estimated' ist kein Modifikator
The <>_<> relatively <>_<> low <Qv> <R10> death <St> <R5,R10> toll <*> <>_<> in <>_<>
Chile <>_<> ( <>_<> estimated <>_<> at <>_<> 6,000 <Qk> <R5> ) <>_<> is <>_<> explained <>_<>
in <>_<> part <>_<> by <>_<> the <>_<> low <>_<> population <>_<> density <>_<> and <>_<> by <>_<>
buildings <>_<> being <>_<> built <>_<> taking <>_<> into <>_<> account <>_<> that <>_<> the <>_<>
region <>_<> is <>_<> very <>_<> active <>_<> geologically <>_<> . <>_<>
```

Insbesondere gilt das auch für `feared`:

```
## 'feared' ist kein Modifikator, die Unischerheit in der Aussage wird nicht
mit-annotiert
More <M> <R3> than <*> <>_<> 100 <Qk> <R3> people <O> <R3> in <>_<> his <>_<> village <>_<>
of <>_<> 1,000 <>_<> were <>_<> feared <>_<> dead <St> <R3> , <>_<> he <>_<> said <>_<> . <>_<>
```

B.2.1.4 Schadensindikator

Diese Entität setzt sich aus den Token zusammen, welche den Schaden an Menschen anzeigen. Auch sie wird weiter unterteilt.

Kommen in einem Satz/Dokument mehrere Kandidaten als Indikator in Frage, so muss einer davon ausgewählt werden (siehe Beispiel unter Abschnitt B.2.1.4.2). Welcher das ist wird durch den Annotator entschieden. Als Richtlinie sollte gelten, welcher SCHADENSINDIKATOR beim Lesen den deutlicheren/dominanteren/offensichtlicheren Eindruck hinterlassen hat. Ebenso sollte man sich an der Menge der bisher gewählten Indikatoren orientieren und sich im Zweifelsfall für einen schon an anderer Stelle erwählten SCHADENSINDIKATOR entscheiden.

B.2.1.4.1 Getötet

Damit sind Indikatoren für Tote zu annotieren.

Tag: `St` (wie Schadensindikator getötet)
Beispiele: `bodies, causalities, crushing, dead, death, death toll, died, fatalities, killed, lost, number dead, number of bodies, number of deaths, body, claimed`

```
## 'deaths' zeigt die Art des Schadens an
The <O> 1985 <O> Mexico <O> City <O> earthquake <O> , <O> a <O> magnitude <O>
8.1 <O> earthquake <O> that <O> struck <O> Mexico <O> on <O> 19 <O> September <O>
1985 <O> at <O> 7:19 <O> local <O> time <O> , <O> caused <O> the <O>
deaths <St> <R0> of <O> about <M> <R0> 10,000 <Qk> <R0> people <O> <R0> and <O>
serious <O> damage <O> in <O> the <O> nation <O> 's <O> capital <O> . <O>
```

Auch Ausdrücke, in denen keine Schlagwörter wie `death`, `body`, ... vorkommen, aber eindeutig vom Tod von Personen die Rede ist, sind zu annotieren:

```
## 'overwhelmed by the waves' -> tot
Many <Qv> <R4> of <O> Scilla <O> 's <O> residents <O> <R4> , <O> frightened <O>
by <O> the <O> tremors <O> of <O> the <O> previous <O> day <O> had <O> moved <O>
onto <O> the <O> open <O> beach <O> for <O> the <O> night <O> , <O> where <O>
they <O> were <O> overwhelmed <St> <R4> by <*> <O> the <*> <O> waves <*> <O> . <O>
```

Ist der Schadensindikator über mehrere, nichtbenachbarte Tokens aufgeteilt, sollte darauf geachtet werden, dass dasjenige Wort als `St` annotiert wird, das notwendig ist, um die Relationsinstanz als Schaden zu erkennen. Andere Tokens sind in diesen Fällen oft sinnvoll als Objekt aufzufassen:

```
## Ohne 'claimed' würde das Tupel keinen klaren Schadensindikator enthalten
Severe <O> , <O> continuous <O> flooding <O> in <O> Thailand <O> has <O>
claimed <St> <R1> 32 <Qk> <R1> lives <O> <R1> , <O> Thailand <O> 's <O> official <O>
news <O> agency <O> reported <O> Saturday <O> . <O>
```

B.2.1.4.2 Verletzt

Damit sind Indikatoren für Verletzte zu annotieren.

Tag: `Sl` (wie Schadensindikator verletzt)
Beispiele: `injured, injuries, injuring, number of injured`

```
## 'injured' zeigt die Art des Schadens an
## 'received treatment' und 'needed hospitalization' sind weitere Indikatoren,
welche zugunsten 'injured' nicht gewählt wurden
9,600 <Qk> <R17> injured <Sl> <R17,R18> people <O> <R17,R18> received <O>
treatment <O> , <O> including <O> 1,879 <Qk> <R18> who <O> needed <O>
hospitalization <O> . <O>
```

B.2.1.4.3 Verschüttet

Damit sind Indikatoren für Verschüttete zu annotieren.

Tag: `Ss` (wie Schadensindikator verschüttet)
Beispiele: `buried` (im Sinne von *verschüttet*), `trapped, trapping`

```
## 'trapped' zeigt die Art des Schadens an
Those <O> who <O> were <O> rescued <O> first <O> were <O> taken <O> to <O>
another <O> building <O> for <O> treatment <O> , <O> as <O> the <O>
ambulances <O> <R21> were <O> trapped <Ss> <R21> inside <O> the <O> collapsed <O>
tower <O> . <O>
```

Hinter `buried` kann vieles stecken: tot, verletzt, lebendig. Solange nichts anderes indiziert wird, ist `Ss` als Annotation zu wählen.

B.2.1.4.4 Vermisst

Damit sind Indikatoren für Vermisste zu annotieren.

Tag: `Sm` (wie Schadensindikator vermisst)
Beispiele: `missing, track down, unaccounted for`

```
## 'track down' zeigt die Art des Schadens an
Canadian_<>_<> officials_<>_<> continue_<>_<> to_<>_<> try_<>_<> to_<>_<> track_<Sm>_<R1>
down_<*>_<> 337_<Qk>_<R1> Canadians_<O>_<R1> in_<>_<> Chile_<>_<> following_<>_<> an_<>_<>
earthquake_<>_<> on_<>_<> Saturday_<>_<> ,_<>_<> Foreign_<>_<> Affairs_<>_<> Minister_<>_<>
Lawrence_<>_<> Cannon_<>_<> said_<>_<> Tuesday_<>_<> ._<>_<>
```

B.2.1.4.5 Obdachlos

Damit sind Indikatoren für Obdachlose/Heimatlose zu annotieren.

Tag: `So` (wie Schadensindikator obdachlos)
Beispiele: `homeless, lost their homes`

```
## 'homeless' zeigt die Art des Schadens an
It_<>_<> has_<>_<> been_<>_<> estimated_<>_<> that_<>_<> about_<>_<> 40_<>_<> %_<>_<> of_<>_<>
the_<>_<> houses_<>_<> in_<>_<> Valdivia_<>_<> were_<>_<> destroyed_<>_<> ,_<>_<> leaving_<>_<>
20,000_<Qk>_<R8> people_<O>_<R8> homeless_<So>_<R8> ._<>_<>
```

Schäden an Gebäuden sind – obwohl sie letztlich Menschen obdachlos machen können – nicht zu annotieren, da sie keine Schäden an Menschen darstellen.

B.2.1.4.6 Betroffen

Damit sind Indikatoren für Betroffene zu annotieren.

Tag: `Sb` (wie Schadensindikator betroffen)
Beispiele: `affected, flood-affected`

```
## 'affected' zeigt die Art des Schadens an
Interruption_<>_<> of_<>_<> classes_<>_<> ,_<>_<> either_<>_<> to_<>_<> the_<>_<> lack_<>_<>
of_<>_<> facilities_<>_<> and/or_<>_<> the_<>_<> need_<>_<> to_<>_<> help_<>_<> with_<>_<>
rescue_<>_<> efforts_<>_<> ,_<>_<> affected_<Sb>_<R3> over_<M>_<R3> 1.5_<Qk>_<R3>
million_<*>_<> students_<O>_<R3> ._<>_<>
```

Es werden nur explizite, d.h. wörtliche, Formen von `affected` annotiert. Alles was hinter dem Begriff „betroffen“ stecken könnte wird ignoriert.

```
## 'number of jobs lost' ist kein Sb
The_<>_<> number_<>_<> of_<>_<> jobs_<>_<> lost_<>_<> due_<>_<> to_<>_<> the_<>_<> event_<>_<>
was_<>_<> estimated_<>_<> at_<>_<> 200,000_<>_<> ._<>_<>

## 'waiting for relocation' ist kein Sb
As_<>_<> of_<>_<> 2005_<>_<> ,_<>_<> there_<>_<> are_<>_<> still_<>_<> two_<>_<> camps_<>_<>
where_<>_<> approximately_<>_<> eighty_<>_<> families_<>_<> are_<>_<> still_<>_<> waiting_<>_<>
for_<>_<> relocation_<>_<> from_<>_<> the_<>_<> earthquake_<>_<> ._<>_<>
```

B.2.1.4.7 Evakuiert Damit sind Indikatoren für Evakuierte zu annotieren.

Tag: `Se` (wie Schadensindikator evakuiert)
Beispiele: `evacuated, evacuation, evacuation center, evacuee`

```
## 'evacuated' zeigt die Art des Schadens an
In_<>_<> the_<>_<> province_<>_<> of_<>_<> Hunan_<>_<> ,_<>_<> which_<>_<> neighbours_<>_<>
Guizhou_<>_<> ,_<>_<> around_<M>_<R11> 16,000_<Qk>_<R11> people_<O>_<R11> were_<>_<>
evacuated_<Se>_<R11> in_<>_<> Loudi_<>_<> city_<>_<> during_<>_<> rainstorms_<>_<> ,_<>_<>
Xinhua_<>_<> reported_<>_<> on_<>_<> Monday_<>_<> ._<>_<>
```

Analog zu „betroffen“ werden nur explizite, d.h. wörtliche, Formen von `evacuated` annotiert. Alles was man ebenfalls als eine Art von Evakuierung ansehen könnte wird ignoriert.

```
## 'leave their homes' ist kein Se
More_<>_<> than_<>_<> 13,000_<>_<> people_<>_<> there_<>_<> have_<>_<> been_<>_<> forced_<>_<>
to_<>_<> leave_<>_<> their_<>_<> homes_<>_<> ._<>_<>
```

Derivate oder Komposita von/mit `evacuation`/`evacuated` sind allerdings ebenfalls zu annotieren:

```
## 'evacuation center' ist eindeutiger Se
About_<M>_<R70> 88,000_<Qk>_<R70> people_<O>_<R70> were_<>_<> being_<>_<> served_<>_<>
at_<>_<> evacuation_<Se>_<R70> centers_<*>_<> ._<>_<>
```

Ebenso aktive Formen:

```
## Aktives evacuate
About_<M>_<R2> 800_<Qk>_<R2> residents_<O>_<R2> have_<>_<> evacuated_<Se>_<R2> to_<>_<>
higher_<>_<> ground_<>_<> ,_<>_<> MCOT_<>_<> said_<>_<> ._<>_<>
```

Es werden allerdings nur Evakuierungen annotiert, deren Objekte Personen sind (siehe Hinweis # 3). Ist von einer Evakuierung von Häusern/Haushalten die Rede, sind nämlich insbesondere die Quantiatätsangaben nicht einfach auf Menschen übertragbar:

```
## evacuation of homes nicht annotiert, weil kein Bericht von Personenschäden
Cornwall_<>_<> floods_<>_<> force_<>_<> evacuation_<>_<> of_<>_<> 100_<>_<> homes_<>_<>
```

B.2.1.4.8 Kombinationen Sollte der beschriebene Schaden aus einer Kombination[4] von obigen Schäden bestehen, so ist er als solcher in Form einer Multi-Token-Entität zu annotieren. Dazu werden die entsprechenden `S`-Tags per Komma getrennt aufgelistet. Die Tagreihenfolge richtet sich nach der Nennreihenfolge in der Entität.

```
## 'killed or listed as missing' zeigt die Art des Schadens an und kann
sinnerhaltent nur als Kombination St,Sm annotiert werden; man beachte die
Tagreihenfolge
More_<M>_<R6> than_<*>_<> 170,000_<Qk>_<R6> Indonesians_<O>_<R6> were_<>_<>
killed_<St,Sm>_<R6> or_<*>_<> listed_<*>_<> as_<*>_<> missing_<*>_<> after_<>_<> a_<>_<>
9.15_<>_<> magnitude_<>_<> earthquake_<>_<> off_<>_<> Indonesia_<>_<> 's_<>_<> Aceh_<>_<>
province_<>_<> on_<>_<> Sumatra_<>_<> island_<>_<> triggered_<>_<> a_<>_<> tsunami_<>_<> in_<>_<>
December_<>_<> 2004_<>_<> ._<>_<>
```

Dies gilt auch in Fällen, bei denen eine sinngemäße Annotation in Form einer Kombination von Relationsinstanzen möglich wäre.

[4] Dies kann einer ODER- oder UND-verknüpfte Kombination sein.

```
## 'injured or made homeless' zeigt die Art des Schadens an und wird ebenfalls als
Kombination annotiert, obwohl hier theoretisch eine sinngemäße Annotation mit zwei
Relationsinstanzen möglich wäre (countless injured + countless homeless)
Almost <M> <R12> 87,000 <Qk> <R12> people <O> <R12,R13> have <>_<> died <St> <R12>
during <>_<> that <>_<> time <>_<> ,_<>_<> and <>_<> countless <Qv> <R13> more <>_<> have <>_<>
been <>_<> injured <Sl,So> <R13> or <*>_<> made <*>_<> homeless <*>_<> ._<>_<>
```

B.2.1.5 Negation

Sollte die korrekte Wiedergabe eines Schadens eine Verneinung benötigen, so ist diese Entität zu verwenden.

Tag: `N` (wie N egation)

Beispiele: `not, n't`

```
## 'not' als Negation ist für die unverfälschte Wiedergabe der Relationsinstanz
notwendig
These <>_<> landslides <>_<> did <>_<> not <N> <R6> cause <>_<> many <Qv> <R6>
fatalities <StE> <R6> nor <>_<> significant <>_<> economical <>_<> losses <>_<> because <>_<>
most <>_<> of <>_<> the <>_<> areas <>_<> were <>_<> uninhabited <>_<> with <>_<> only <>_<>
minor <>_<> roads <>_<> ._<>_<>
```

Die Verneinung bezieht sich dabei immer auf die (möglicherweise modifizierte) Quantität, nicht nur auf den Schadensindikator. Entsprechend ist folgendes Beispiel *keine* Negation, weil hier nicht die Rede davon ist, dass "es nicht der Fall ist, dass (`M`) `Q` (`O`) `S`", sondern dass "(`M`) `Q` (`O`) nicht-`S`":

```
## 'not' verneint nicht die gesamte Quantitaet/Aussage
Ramos <>_<> said <>_<> despite <>_<> government <>_<> warning <>_<> ,_<>_<> some <>_<> did <>_<>
not <>_<> evacuate <>_<> ._<>_<>
```

Der Spezialfall von nachträglich korrigierten oder zurückgenommenen Angaben wird nicht als Negation, sondern als normale Relationsinstanz annotiert, gleichzeitig mit (falls vorhanden) der Korrektur:

```
## Zwei Relationsinstanzen
The <>_<> number <St> <R0,R1> of <*>_<> deaths <*>_<> initially <>_<> reported <>_<> by <>_<>
the <>_<> Chinese <>_<> government <>_<> was <>_<> 655,000 <Qk> <R0> ,_<>_<> but <>_<> this <>_<>
number <>_<> has <>_<> since <>_<> been <>_<> stated <>_<> to <>_<> be <>_<> around <M> <R1>
240,000 <Qk> <R1> ._<>_<>
```

B.2.2 Relationsinstanzen

Relationsinstanzen sind über Entitäten in einem Dokument definiert und können satzübergreifend sein. Der Instanzname ist frei wählbar, muss aber innerhalb eines Dokuments eindeutig sein. Wenn eine Entität an mehreren Instanzen beteiligt ist, so sind die Namen mit Komma zu trennen (siehe Beispiel unter B.2.1.2.1).

Jedes Relationstupel besteht mindestens aus einer Entität SCHADENSINDIKATOR. Sollte ein MODIFIKATOR Teil einer Relationsinstanz sein, so muss auch eine QUANTITÄT vorhanden sein (siehe B.2.1.3). Jeder Entitätstyp kommt höchstens einmal pro Tupel vor.

Einige Varianten zur Veranschaulichung:

```
## vollständige Relationsinstanz
...almost <M> <R2> 70,000 <Qk> <R2> people <O> <R2> injured <S1> <R2>...

## ohne Modifikator
...70,000 <Qk> <R2> people <O> <R2> injured <S1> <R2>...

## ohne Objekt
...many <Qv> <R2> injured <S1> <R2>...

## nur Schadensindikator
...caused <> <> injured <S1> <R2>...
```

Es gibt also auch Relationstupel der Größe 1.

Tupel können zueinander in Beziehung stehen, z. B. `25 people were injured, including 10 tourists`. Solche Konstrukte sind normal als zwei Tupel zu annotieren (siehe auch Beispiel unter B.2.1.2.1).

```
## die Beziehung 'including' zwischen R2 und R3 wird ignoriert
25 <Qk> <R2> people <O> <R2> were <> <> injured <S1> <R2,R3> , <> <> including <> <>
10 <Qk> <R3> tourists <O> <R3> . <> <>
```

Für Bereichsangaben existiert eine verkürzte Annotationsform, siehe B.2.1.2.5. Das gilt nur, wenn jeweils absolute Quantitäten genannt werden; Instanzen mit relativen Quantitäten werden in der Regel nicht annotiert, siehe B.2.1.2.4.

B.2.2.1 1-Tupel

Wie oben gezeigt gibt es auch Instanzen von ERDBEBENSCHADEN der Größe 1. Nun ist aber nicht jeder allein stehende potenzielle SCHADENSINDIKATOR als 1-Tupel aufzufassen, wie folgendes Beispiel verdeutlichen soll:

```
## keine Relationsinstanz
The <> <> death <> <> toll <> <> would <> <> have <> <> been <> <> much <> <> higher <> <>...
```

Als ERDBEBENSCHADEN ist immer dann zu annotieren, wenn über konkrete Schäden/Opfer berichtet wird (vgl. Beispiel bei der Veranschaulichung oben). Wird nur abstrakt auf Schäden Bezug genommen (siehe letztes Beispiel), so handelt es sich um keine Instanz von ERDBEBENSCHADEN. Hier muss der Annotator entscheiden, welcher Fall vorliegt.

B.2.2.2 Satzübergreifend

Wie beschrieben können Relationstupel auch satzübergreifend sein. Dies betrifft hauptsächlich den Bezug auf das OBJEKT.

Wenn wir Meldungen über Erdbeben lesen, dann ist das OBJEKT „Mensch“ (`people`) in unserem Bewusstsein schon voraktiviert. Wir tendieren also dazu, Sätze welche eine Relationsinstanz ERDBEBENSCHADEN ohne OBJEKT enthalten, implizit mit „Mensch“ zu verbinden (`10,000 were killed` wird vervollständigt zu `10,000` *people* `were killed`).

Genauso kommt es vor, dass am Anfang des Textes das OBJEKT genannt wird und später auf eine explizite Wiederholung verzichtet wird. Im Falle von `people` stellt sich dann die Frage: Verweist diese spätere Relationsinstanz auf das voraktivierte „Mensch“ oder auf die frühere Nennung?

Bei Ersterem wird kein OBJEKT annotiert und das Tupel enthält keines. Bei Zweiterem wird das frühere OBJEKT in das Tupel eingebunden und es wird dadurch satzübergreifend. Die Entscheidung, welcher Fall gilt, muss vom Annotator getroffen werden.

B.2.3 Hinweise

#0 Die Groß-/Kleinschreibung der Tags und Relationstupelnamen ist egal.

#1 Es werden nur reale Schäden annotiert, also keine Was-wäre-wenn-Konstrukte.

```
## 'hundreds of deaths' ist kein Schaden
Modern_<>_<> studies_<>_<> estimate_<>_<> that_<>_<> if_<>_<> a_<>_<> similar_<>_<> quake_<>_<>
shook_<>_<> Boston_<>_<> today_<>_<> ,_<>_<> it_<>_<> would_<>_<> result_<>_<> in_<>_<> as_<>_<>
much_<>_<> as_<>_<> $_<>_<> 5_<>_<> billion_<>_<> in_<>_<> damage_<>_<> and_<>_<> hundreds_<>_<>
of_<>_<> deaths_<>_<> ._<>_<>
```

Insbesondere wird auch bei Evakuierungen nicht annotiert, wenn nur über Anordnung, nicht aber über ihre Durchführung berichtet wird:

```
## 'ordered evacuations' wird nicht annotiert!
With_<>_<> more_<>_<> rains_<>_<> forecast_<>_<> ,_<>_<> Brazilian_<>_<> authorities_<>_<>
have_<>_<> ordered_<>_<> evacuations_<>_<> for_<>_<> at_<>_<> least_<>_<> 5,000_<>_<>
families_<>_<> living_<>_<> in_<>_<> especially_<>_<> perilous_<>_<> areas_<>_<> in_<>_<>
Rio_<>_<> de_<>_<> Janeiro_<>_<> state_<>_<> ,_<>_<> Agencia_<>_<> Brasil_<>_<> said_<>_<> ._<>_<>
```

Mittelwerte über Zeiträume sind allerdings reale Schäden und werden entsprechend annotiert:

```
## averages werden annotiert
Floods_<>_<> in_<>_<> India_<>_<> kill_<St>_<R1> 1,793_<Qk>_<R1> people_<O>_<R1> each_<>_<>
year_<>_<> ,_<>_<> on_<>_<> an_<>_<> average_<>_<> ,_<>_<> and_<>_<> cause_<>_<> losses_<>_<> of_<>_<>
$_<>_<> 575_<>_<> million_<>_<> each_<>_<> year_<>_<> ,_<>_<> including_<>_<> damaging_<>_<>
crops_<>_<> in_<>_<> 3.7_<>_<> million_<>_<> hectares_<>_<> ._<>_<>
```

#2 Es werden nur Schäden annotiert, welche mittel- und langfristig gelten. Wenn also Menschen während des Bebens in Panik auf die Straße rennen, so ist das an sich kein ERDBEBENSCHADEN.[5]

[5] Kommentar des Autors: Durch die engere `Sb`-Definition seit v0.2 ist dieser Hinweis obsolet.

#3 Wie eingangs erwähnt geht es um die Erkennung von Schäden an Menschen, nicht an Gebäuden oder ähnlichem. D. h. eine Angabe wie `About 1.5 million homes have been damaged.` ist kein ERDBEBENSCHADEN im Rahmen dieser Arbeit, obwohl klar ist, dass dadurch auch potenziell Menschen betroffen sind. Diese impliziten Schäden bilden keinen Bestandteil dieser Arbeit.

#4 Es findet keine Anaphernresolution[6] statt. Die Anapher wird als OBJEKT annotiert und nicht das Antezedens.

```
## 'She' ist eine Anapher
She_<O>_<R8> was_<>_<> injured_<S1>_<R8> ...
```

#5 Es werden nur Schäden annotiert, welche zeitlich/örtlich unmittelbar mit einem Beben/einer Flut zusammen hängen. Wenn also z. B. bei Aufräumarbeiten jemand verletzt wird, so ist das kein zu annotierender Schaden. Tote z.B. durch eine Panik, die durch die Katastrophe ausgelöst wurde und gleichzeitig auftritt, zählen als Schäden. Werden im Text andere Erdbeben/Fluten erwähnt neben dem hauptsächlich beschriebenen, sind ihre Schäden genauso zu annotieren! Tote durch Seuchen oder Krankheiten nach einer Katastrophe werden schon als zu indirekte Schäden betrachtet und nicht annnotiert. Tote durch Blitzschlag sind zwar unmittelbar gleichzeitig mit Regen/Flutkatastrophen, aber ursächlich nicht verbunden, werden also nicht annotiert.

#6 Für Multi-Token-Entitäten gilt: Nur das erste Token bekommt die Annotation, alle weiteren Token ein _<*>_<>.
Grundregel: Soviel wie nötig, so wenig wie möglich.

- ohne Artikel
  - `death toll` statt `the death toll`
- ohne Adjektiv (wenn nicht bedeutungstragend)
  - `man` statt `20-year-old man`
- bei zusammengesetzten Nomen nur den bedeutungstragenden Kopf (Oberbegriff)
  - `teachers` statt `high school teachers`
  - auch bei Kompositionen, bei denen entscheidende Bedeutung außerhalb des Kopfs liegt: `members` statt `family members`

  dabei aber feste Verbindungen beibehalten
  - `number of deaths` statt nur `deaths`
- bei Verben ohne Hilfverb
  - nur `injured` statt `were injured`

#7 Andere Effekte auf Menschen, wie z. B. `More than 100 prisoners escaped from a jail.` sind zu ignorieren.

[6] http://www.glottopedia.de/index.php/Anaphernresolution

C Detailed Extraction Results

Table C.1: Detailed evaluation results for oracle NER, including the pipeline configurations. BinRE denotes the results for the intermediate entity pairs (if applicable).

Corpus		Method	RE Config (active tweaks)	Step	FP/TP/FN	P/R/F1
Earthquake	Wikipedia	SVM_{MC}	IgnoreEntityPairDirection IgnoreEntityPairType Enum-Filter	BinRE RE	24/763/72 26/262/49	.970/.914/.941 .910/.842/.875
		SVM_{WC}	IgnoreEntityPairDirection IgnoreEntityPairType Enum-Filter	RE	38/261/50	.873/.839/.856
		PM	IgnoreDependencyDirection IgnoreDependencyType Enum-Filter	BinRE RE	88/727/108 27/233/78	.892/.871/.881 .896/.749/.816
		TN	Enum-Filter	BinRE RE	185/818/17 71/256/55	.816/.980/.890 .783/.823/.803
		DARE	IgnoreDependencyType Enum-Filter	RE	35/219/92	.862/.704/.775
Earthquake	News	SVM_{MC}	IgnoreEntityPairDirection Enum-Filter	BinRE RE	45/1244/58 32/396/45	.965/.955/.960 .925/.898/.911
		TN	Enum-Filter	BinRE RE	180/1295/7 54/399/42	.878/.995/.933 .881/.905/.893
		SVM_{WC}	Enum-Filter	RE	59/398/43	.871/.902/.886
		PM	IgnoreDependencyDirection IgnoreDependencyType Enum-Filter	BinRE RE	66/1197/105 27/372/69	.948/.919/.933 .932/.844/.886
		DARE	IgnoreDependencyType Enum-Filter	RE	28/358/83	.927/.812/.866
Flood	News	SVM_{WC}	IgnoreEntityPairDirection IgnoreEntityPairType Enum-Filter	RE	41/638/68	.940/.904/.921
		PM	IgnoreDependencyType Enum-Filter	BinRE RE	33/1954/203 32/601/105	.983/.906/.943 .949/.851/.898
		SVM_{MC}	IgnoreEntityPairDirection Enum-Filter	BinRE RE	18/2001/156 52/609/97	.991/.928/.958 .921/.863/.891
		DARE	IgnoreDependencyType Enum-Filter	RE	60/558/148	.903/.790/.843
		TN	Enum-Filter	BinRE RE	324/2128/29 138/611/95	.868/.987/.923 .816/.865/.840

FP: false positives; TP: true positives; FN: false negatives
P: precision; R: recall

Table C.2: Detailed evaluation results for non-oracle NER, including the pipeline configurations. BinRE denotes the results for the intermediate entity pairs (if applicable).

Corpus		Method	RE Config (active tweaks)	NER + Config	Step	FP/TP/FN	P/R/F1
Earthquake	Wikipedia	SVM_{MC}	Enum-Filter	DctRgx with M-Filter	NER	3556/743/54	.173/.932/.292
					BinRE	219/669/166	.753/.801/.777
					RE	101/210/101	.675/.675/.675
		SVM_{WC}	IgnoreEntityPairDirection IgnoreEntityPairType Enum-Filter	CRF with Training-Filter WoMatch-Filter	NER	618/697/100	.530/.875/.660
					RE	95/201/110	.679/.646/.662
		DARE	IgnoreDependencyDirection IgnoreDependencyType Enum-Filter	DctRgx with M-Filter A-Filter	NER	2559/729/68	.222/.915/.357
					RE	86/195/116	.694/.627/.659
		PM	IgnoreDependencyDirection IgnoreDependencyType Enum-Filter	DctRgx with M-Filter A-Filter	NER	2559/729/68	.222/.915/.357
					BinRE	369/637/198	.633/.763/.692
					RE	87/193/118	.689/.621/.653
		TN	Enum-Filter	CRF with Training-Filter WoMatch-Filter	NER	618/697/100	.530/.875/.660
					BinRE	452/674/161	.599/.807/.687
					RE	167/193/118	.536/.621/.575
Earthquake	News	PM	IgnoreDependencyType Enum-Filter	DctRgx with M-Filter A-Filter	NER	1557/1139/54	.422/.955/.586
					BinRE	304/1106/196	.784/.849/.816
					RE	90/321/120	.781/.728/.754
		SVM_{MC}	IgnoreEntityPairDirection UseGoldEntities Enum-Filter	CRF with Training-Filter WoMatch-Filter	NER	436/1100/93	.716/.922/.806
					BinRE	239/1137/165	.826/.873/.849
					RE	105/330/111	.759/.748/.753
		DARE	IgnoreDependencyType Enum-Filter	DctRgx with M-Filter A-Filter	NER	1557/1139/54	.422/.955/.586
					RE	90/320/121	.780/.726/.752
		SVM_{WC}	Enum-Filter	DctRgx with M-Filter A-Filter	NER	1557/1139/54	.422/.955/.586
					RE	131/331/110	.716/.751/.733
		TN	Enum-Filter	CRF with WoMatch-Filter	NER	84/1021/172	.924/.856/.889
					BinRE	238/1081/221	.820/.830/.825
					RE	100/292/149	.745/.662/.701
Flood	News	PM	IgnoreDependencyType Enum-Filter	CRF with Training-Filter WoMatch-Filter	NER	972/1832/118	.653/.939/.771
					BinRE	415/1806/351	.813/.837/.825
					RE	123/533/173	.812/.755/.783
		SVM_{WC}	IgnoreEntityPairDirection IgnoreEntityPairType Enum-Filter	CRF with Training-Filter WoMatch-Filter	NER	972/1832/118	.653/.939/.771
					RE	170/542/164	.761/.768/.764
		SVM_{MC}	IgnoreEntityPairDirection IgnoreEntityPairType Enum-Filter	CRF with Training-Filter	NER	981/1834/116	.652/.941/.770
					BinRE	309/1830/327	.856/.848/.852
					RE	170/530/176	.757/.751/.754
		DARE	IgnoreDependencyType Enum-Filter	CRF with Training-Filter WoMatch-Filter	NER	972/1832/118	.653/.939/.771
					RE	140/501/205	.782/.710/.744
		TN	Enum-Filter	CRF with WoMatch-Filter	NER	174/1712/238	.908/.878/.893
					BinRE	443/1819/338	.804/.843/.823
					RE	206/480/226	.700/.680/.690

FP: false positives; TP: true positives; FN: false negatives
P: precision; R: recall

D List of Events

These are the earthquake and flood events covered in the case studies in Chapter 5, i.e. the English Wikipedia articles.

Date Day of the event start as defined in the gold standard (UTC).

Location Toponyms used at the infobox.

Casualties Number of casualties reported at the infobox.

Updates Number of updates of the casualties reported at the infobox.

Update@80% Time elapsed when 80 % of the final casualty numbers are reported at the infobox. This time corresponds to the duration of the event (floods) or rescue operations (earthquakes, floods). We use 80 % as final numbers are sometimes posted months or years after the event, biasing the value for time elapsed.

Training Events

Date	Location	Casualties	Updates	Update@80 %
2006-07-17[1]	South of Java, Indonesia	665	17	2.33 d
2007-08-15[2]	Chincha Alta, Peru	519	28	0.788 d
2008-05-12[3]	Sichuan Province, China	69 197	88	160 d
2008-10-05[4]	Eastern Kyrgyzstan	75	6	1.14 d
2009-04-06[5]	L'Aquila, Abruzzo, Italy	294	36	2.20 d
2009-09-30[6]	Southern Sumatra, Indonesia	1115	7	1.29 d
2010-04-13[7]	Yushu, Qinghai, China	2698	18	8.05 d
2011-03-10[8]	Yunnan, China	26	13	0.357 d
2011-10-23[9]	Van, Turkey	604	33	4.04 d
2012-05-20[10]	Emilia-Romagna, Italy	27	22	9.46 d

[1] https://en.wikipedia.org/wiki/July_2006_Java_earthquake
[2] https://en.wikipedia.org/wiki/2007_Peru_earthquake
[3] https://en.wikipedia.org/wiki/2008_Sichuan_earthquake
[4] https://en.wikipedia.org/wiki/2008_Kyrgyzstan_earthquake
[5] https://en.wikipedia.org/wiki/2009_L'Aquila_earthquake
[6] https://en.wikipedia.org/wiki/2009_Sumatra_earthquakes
[7] https://en.wikipedia.org/wiki/2010_Yushu_earthquake
[8] https://en.wikipedia.org/wiki/2011_Yunnan_earthquake
[9] https://en.wikipedia.org/wiki/2011_Van_earthquake
[10] https://en.wikipedia.org/wiki/2012_Northern_Italy_earthquakes

Evaluation Earthquakes

Date	Location	Casualties	Updates	Update@80 %
2006-05-26[11]	Java, Indonesia	5782	28	2.23 d
2007-03-06[12]	Sumatra, Indonesia	67	7	3.01 d
2007-04-01[13]	Solomon Islands	52	11	11.0 d
2007-09-12[14]	Sumatra, Indonesia	25	7	3.14 d
2008-02-03[15]	Lake Kivu, Congo	39	5	1.04 d
2008-06-13[16]	Tōhoku, Japan	13	8	11.7 d
2008-10-28[17]	Northwestern Pakistan	215	6	1.86 d
2009-01-08[18]	Costa Rica	34	6	2.93 d
2009-09-02[19]	Java, Indonesia	79	7	7.77 d
2009-09-29[20]	Samoa Islands	189	7	2.53 d
2010-02-27[21]	Maule, Chile	523	38	31.6 d
2010-03-08[22]	Elâzığ Province, Turkey	42	16	0.105 d
2010-10-25[23]	Sumatra	435	19	3.27 d
2011-02-21[24]	Canterbury, New Zealand	181	52	4.21 d
2011-03-11[25]	Japan	15 870	206	28.3 d
2011-03-24[26]	Shan, Burma	150	13	10.2 d
2011-09-18[27]	Sikkim, India	111	21	1.82 d
2012-02-06[28]	Negros, Cebu, Philippines	113	5	6.49 d
2012-06-11[29]	Hindu kush, Afghanistan	75	5	1.64 d
2012-08-11[30]	Tabriz, Iran	306	6	0.632 d
2012-09-07[31]	Zhaotong, Yunnan, China	81	7	1.05 d
2012-11-07[32]	Guatemala	44	9	0.249 d
2012-11-11[33]	Burma	26	7	2.88 d

[11] https://en.wikipedia.org/wiki/May_2006_Java_earthquake
[12] https://en.wikipedia.org/wiki/March_2007_Sumatra_earthquakes
[13] https://en.wikipedia.org/wiki/2007_Solomon_Islands_earthquake
[14] https://en.wikipedia.org/wiki/September_2007_Sumatra_earthquakes
[15] https://en.wikipedia.org/wiki/2008_Lake_Kivu_earthquake
[16] https://en.wikipedia.org/wiki/2008_Iwate-Miyagi_Nairiku_earthquake
[17] https://en.wikipedia.org/wiki/2008_Pakistan_earthquake
[18] https://en.wikipedia.org/wiki/2009_Costa_Rica_earthquake
[19] https://en.wikipedia.org/wiki/2009_West_Java_earthquake
[20] https://en.wikipedia.org/wiki/2009_Samoa_earthquake
[21] https://en.wikipedia.org/wiki/2010_Chile_earthquake
[22] https://en.wikipedia.org/wiki/2010_El%C3%A2z%C4%B1%C4%9F_earthquake
[23] https://en.wikipedia.org/wiki/October_2010_Sumatra_earthquake_and_tsunami
[24] https://en.wikipedia.org/wiki/2011_Christchurch_earthquake
[25] https://en.wikipedia.org/wiki/2011_T%C5%8Dhoku_earthquake_and_tsunami
[26] https://en.wikipedia.org/wiki/2011_Burma_earthquake
[27] https://en.wikipedia.org/wiki/2011_Sikkim_earthquake
[28] https://en.wikipedia.org/wiki/2012_Visayas_earthquake
[29] https://en.wikipedia.org/wiki/June_2012_Afghanistan_earthquakes
[30] https://en.wikipedia.org/wiki/2012_East_Azerbaijan_earthquakes
[31] https://en.wikipedia.org/wiki/2012_Yunnan_earthquakes
[32] https://en.wikipedia.org/wiki/2012_Guatemala_earthquake
[33] https://en.wikipedia.org/wiki/2012_Shwebo_earthquake

Evaluation Floods

Date	Location	Casualties	Updates	Update@80 %
2008-11-22[34]	Santa Catarina, Brazil	128	9	7.00 d
2009-04-22[35]	Maranhão, Brazil	44	6	18.0 d
2009-10-01[36]	Messina, Sicily, Italy	31	8	5.99 d
2010-04-06[37]	Rio de Janeiro, Brazil	249	9	3.69 d
2010-07-21[38]	Pakistan	1781	10	16.7 d
2011-01-12[39]	Rio de Janeiro, Brazil	903	32	7.77 d
2011-06-06[40]	China	355	8	42.4 d
2011-08-03[41]	Thailand	815	15	122 d
2011-08-11[42]	Sindh, Pakistan	434	5	41.0 d

[34] https://en.wikipedia.org/wiki/2008_Santa_Catarina_floods
[35] https://en.wikipedia.org/wiki/2009_Brazilian_floods_and_mudslides
[36] https://en.wikipedia.org/wiki/2009_Messina_floods_and_mudslides
[37] https://en.wikipedia.org/wiki/April_2010_Rio_de_Janeiro_floods_and_mudslides
[38] https://en.wikipedia.org/wiki/2010_Pakistan_floods
[39] https://en.wikipedia.org/wiki/January_2011_Rio_de_Janeiro_floods_and_mudslides
[40] https://en.wikipedia.org/wiki/2011_China_floods
[41] https://en.wikipedia.org/wiki/2011_Thailand_floods
[42] https://en.wikipedia.org/wiki/2011_Sindh_floods

Bibliography

[1] N. Afzal. Complex relations extraction. In *Conference on Language & Technology 2009 (CLT'09)*, 2009.

[2] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94. ACM, 2000.

[3] A. Akbik and A. Löser. Kraken: N-ary facts in open information extraction. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 52–56. ACL, 2012.

[4] D. E. Alexander. Social media in disaster risk reduction and crisis management. *Science and Engineering Ethics*, 20(3):717–733, 2014.

[5] O. Alonso, J. Strötgen, R. A. Baeza-Yates, and M. Gertz. Temporal information retrieval: Challenges and opportunities. *Twaw*, 11:1–8, 2011.

[6] X. Amatriain. More data or better models?, July 2012. `http://technocalifornia.blogspot.de/2012/07/more-data-or-better-models.html` (accessed 2014-10-29).

[7] J. Aslam, F. Diaz, M. Ekstrand-Abueg, R. McCreadie, V. Pavlu, and T. Sakai. Trec 2015 temporal summarization track overview. Technical report, DTIC Document, 2016.

[8] N. Bach and S. Badaskar. A survey on relation extraction. *LTI, Carnegie Mellon University*, 2007.

[9] M. Banko, M. Cafarella, et al. Open information extraction from the web. In *IJCAI*, volume 7, 2007.

[10] L. Barbosa and J. Freire. Siphoning hidden-web data through keyword-based interfaces. In *SBBD*, pages 309–321, 2004.

[11] K. Berberich, S. Bedathur, O. Alonso, and G. Weikum. A language modeling approach for temporal information needs. In *European Conference on Information Retrieval*, pages 13–25. Springer, 2010.

[12] M. K. Bergman. White paper: the deep web: surfacing hidden value. *Journal of electronic publishing*, 7(1), 2001.

[13] J. Bhogal, A. Macfarlane, and P. Smith. A review of ontology based query expansion. *Information processing & management*, 43(4):866–886, 2007.

[14] A. Bosch, T. Bogers, and M. Kunder. Estimating search engine index size variability: a 9-year longitudinal study. *Scientometrics*, 107(2):839–856, 2016.

[15] S. Brin. Extracting patterns and relations from the world wide web. In *In WebDB Workshop at EDBT'98*, pages 172–183, 1998.

[16] R. Bunescu and R. Mooney. A shortest path dependency kernel for relation extraction. In *HLT'05*, 2005.

[17] R. Campos, G. Dias, A. M. Jorge, and A. Jatowt. Survey of temporal information retrieval and related applications. *ACM Computing Surveys (CSUR)*, 47(2):15, 2014.

[18] C. Carpineto and G. Romano. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys (CSUR)*, 44(1):1, 2012.

[19] S. Chakrabarti, M. Van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer networks*, 31(11):1623–1640, 1999.

[20] S. Challa, T. Gulrez, Z. Chaczko, and T. Paranesha. Opportunistic information fusion: A new paradigm for next generation networked sensing systems. In *Information Fusion, 2005 8th International Conference on*, volume 1, pages 8–pp. IEEE, 2005.

[21] N. Chambers and D. Jurafsky. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 976–986. Association for Computational Linguistics, 2011.

[22] A. X. Chang and C. Manning. SUTime: A library for recognizing and normalizing time expressions. In *LREC'12*, 2012.

[23] E. Charniak. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[24] E. Charniak and M. Johnson. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[25] Y.-R. Chen, C.-H. Yeh, and B. Yu. Flood damage assessment of an urban area in taiwan. *Natural Hazards*, 83(2):1045–1055, 2016.

[26] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3), 1995.

[27] A. Dalli and Y. Wilks. Automatic dating of documents and temporal text classification. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 17–22. ACL, 2006.

[28] M. Del Vicario, S. Gaito, W. Quattrociocchi, M. Zignani, and F. Zollo. Public discourse and news consumption on online social media: A quantitative, cross-platform analysis of the italian referendum. *arXiv preprint arXiv:1702.06016*, 2017.

[29] L. Döhling and U. Leser. Equatornlp: Pattern-based information extraction for disaster response. In *Terra Cognita 2011*, 2011.

[30] L. Döhling and U. Leser. Extracting and aggregating temporal events from text. In *TempWeb2014*, 2014.

[31] L. Döhling, J. Lewandowski, and U. Leser. A Study in Domain-Independent Information Extraction for Disaster Management. In *Workshop on DIsaster Management and Principled Large-scale information Extraction for and post emergency Logistics*, 2014.

[32] L. Döhling, H. Woith, D. Fahland, and U. Leser. Equator: Faster Decision Making for Geoscientists. In *Proceeding of Workshop on IT support for rescue teams 2011*, 2011.

[33] A. Dong, Y. Chang, Z. Zheng, G. Mishne, J. Bai, R. Zhang, K. Buchner, C. Liao, and F. Diaz. Towards recency ranking in web search. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 11–20. ACM, 2010.

[34] L. Dong and J. Shan. A comprehensive review of earthquake-induced building damage detection with remote sensing techniques. *ISPRS Journal of Photogrammetry and Remote Sensing*, 84:85–99, 2013.

[35] E. C. Dragut, T. Kabisch, C. Yu, and U. Leser. A hierarchical approach to model web query interfaces for web source integration. *Proceedings of the VLDB Endowment*, 2(1):325–336, 2009.

[36] S. Endrullis, A. Thor, and E. Rahm. Entity search strategies for mashup applications. In *ICDE '12*, 2012.

[37] M. R. Endsley. Toward a theory of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):32–64, 1995.

[38] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and M. Mausam. Open Information Extraction: the Second Generation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One*, pages 3–10. AAAI Press, 2011.

[39] D. Fahland, T. M. Gläßer, B. Quilitz, S. Weißleder, and U. Leser. Huodini–flexible information integration for disaster management. In *4th International Conference on Information Systems for Crisis Response and Management(ISCRAM), Delft, NL*, 2007.

[40] D. Farmakiotou, V. Karkaletsis, J. Koutsias, G. Sigletos, C. D. Spyropoulos, and P. Stamatopoulos. Rule-based named entity recognition for greek financial texts. In *In Proceedings of the Workshop on Computational Lexicography and Multimedia Dictionaries (COMLEX 2000)*, pages 75–78, 2000.

[41] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini. The rise of social bots. *Commun. ACM*, 59(7):96–104, June 2016.

[42] M. Filannino, G. Brown, and G. Nenadic. Mantime: Temporal expression identification and normalization in the tempeval-3 challenge. In *SemEval '13*, 2013.

[43] T. Finin, W. Murnane, et al. Annotating named entities in twitter data with crowdsourcing. In *NAACL HLT 2010*, CSLDAMT '10, 2010.

[44] J. R. Finkel, T. Grenager, and C. Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370. Association for Computational Linguistics, 2005.

[45] S. Flaxman, S. Goel, and J. Rao. Filter bubbles, echo chambers, and online news consumption. *Public Opinion Quarterly*, page nfw006, 2016.

[46] K. Fundel, R. Küffner, and R. Zimmer. RelEx - Relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371, 2007.

[47] S. Gauch and J. B. Smith. Search improvement via automatic query reformulation. *ACM Transactions on Information Systems (TOIS)*, 9(3):249–280, 1991.

[48] C. F. Gauss. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium.* Göttingen, 1809.

[49] C. Giuliano, A. Lavelli, and L. Romano. Exploiting Shallow Linguistic Information for Relation Extraction from Biomedical Literature. In *EACL '06*, Trento, Italy, 2006. ACL.

[50] R. Grishman and B. Sundheim. Message Understanding Conference-6: A Brief History. In *COLING*, volume 96, pages 466–471, 1996.

[51] D. Gupta and K. Berberich. Identifying time intervals of interest to queries. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1835–1838. ACM, 2014.

[52] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2nd ed. edition, 4 2006.

[53] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. ACL, 1992.

[54] A. K. Henderson, S. R. Lillibridge, C. Salinas, R. W. Graves, P. B. Roth, and E. K. Noji. Disaster medical assistance teams: providing health care to a community struck by hurricane iniki. *Annals of emergency medicine*, 23(4):726–730, 1994.

[55] C. Hölscher and G. Strube. Web search behavior of internet experts and newbies. *Computer networks*, 33(1):337–346, 2000.

[56] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

[57] M. Imran, C. Castillo, F. Diaz, and S. Vieweg. Processing social media messages in mass emergency: A survey. *ACM Computing Surveys (CSUR)*, 47(4):67, 2015.

[58] M. Inoue and K. Tajima. Noise robust detection of the emergence and spread of topics on the web. In *Proceedings of the 2nd Temporal Web Analytics Workshop*, pages 9–16. ACM, 2012.

[59] H. Isozaki and H. Kazawa. Efficient support vector classifiers for named entity recognition. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[60] N. Jakob and I. Gurevych. Extracting Opinion Targets in a Single- and Cross-domain Setting with Conditional Random Fields. In *EMNLP '10*, 2010.

[61] A. Jatowt, Y. Kawai, and K. Tanaka. Detecting age of page content. In *Proceedings of the 9th annual ACM international workshop on Web information and data management*, pages 137–144. ACM, 2007.

[62] T.-K. Jenssen, A. Lægreid, J. Komorowski, and E. Hovig. A literature network of human genes for high-throughput analysis of gene expression. *Nature genetics*, 28(1):21–28, 2001.

[63] R. J. Kate and R. J. Mooney. Joint entity and relation extraction using card-pyramid parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 203–212. ACL, 2010.

[64] A. Kawtrakul, C. Yingsaeree, and F. Andrès. A framework of nlp based information tracking and related knowledge organizing with topic maps. In *NLDB*, pages 272–283, 2007.

[65] C. Kedzie, K. McKeown, and F. Diaz. Predicting salient updates for disaster summarization. In *ACL (1)*, pages 1608–1617, 2015.

[66] J.-D. Kim, T. Ohta, S. Pyysalo, Y. Kano, and J. Tsujii. Overview of BioNLP'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, pages 1–9. ACL, 2009.

[67] J.-D. Kim, Y. Wang, T. Takagi, and A. Yonezawa. Overview of genia event task in BioNLP shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 7–15. ACL, 2011.

[68] R. Klinger, C. Kolářik, J. Fluck, M. Hofmann-Apitius, and C. M. Friedrich. Detection of IUPAC and IUPAC-like chemical names. *Bioinformatics*, 24(13):i268–i276, 2008.

[69] C. Kohlschütter, P. Fankhauser, and W. Nejdl. Boilerplate detection using shallow text features. In *WSDM '10*, 2010.

[70] W. Kraaij and R. Pohlmann. Viewing stemming as recall enhancement. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '96, pages 40–48, New York, NY, USA, 1996. ACM.

[71] R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, S. Vaithyanathan, and H. Zhu. SystemT: a system for declarative information extraction. *ACM SIGMOD Record*, 37(4):7–13, 2009.

[72] J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML-2001*, 2001.

[73] N. Lao, A. Subramanya, F. Pereira, and W. W. Cohen. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1017–1026. ACL, 2012.

[74] D. B. Leake, A. Maguitman, T. Reichherzer, A. J. Cañas, M. Carvalho, M. Arguedas, and T. Eskridge. Googling from a concept map: Towards automatic concept-map-based query formation. In *Concept maps: Theory, methodology, technology. Proceedings of the first international conference on concept mapping*, volume 1, pages 409–416, 2004.

[75] R. Leaman and G. Gonzalez. Banner: an executable survey of advances in biomedical named entity recognition. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 652–663, 2008.

[76] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 2014.

[77] U. Leser and J. Hakenberg. What makes a gene name? named entity recognition in the biomedical literature. *Briefings in Bioinformatics*, 6(4):357–369, 2005.

[78] X. Li, M. Ge, X. Dai, X. Ren, M. Fritsche, J. Wickert, and H. Schuh. Accuracy and reliability of multi-gnss real-time precise positioning: Gps, glonass, beidou, and galileo. *Journal of Geodesy*, 89(6):607–635, 2015.

[79] C. Liao, T. Chang, M. Krishnaprasad, and M. Bhavsar. Re-ranking search results from an enterprise system, 2010. US Patent App. 12/751,268.

[80] J. L. Lozán and H. Kausch. *Angewandte Statistik für Naturwissenschaftler*. Parey Buchverlag, 1998.

[81] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. An Introduction to Information Retrieval. Cambridge University Press, 2008.

[82] C. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT, 1999.

[83] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[84] M. Marneffe, B. MacCartney, and C. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC-06*, pages 449–454, 2006.

[85] Mausam, M. Schmitz, R. Bart, S. Soderland, and O. Etzioni. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. ACL, 2012.

[86] A. McCallum. MALLET: A Machine Learning for Language Toolkit, 2002.

[87] A. McCallum. Efficiently Inducing Features of Conditional Random Fields. In *UAI '03*, 2003.

[88] R. McDonald, F. Pereira, et al. Simple algorithms for complex relation extraction with applications to biomedical ie. In *ACL '05*, 2005.

[89] P. Menold, R. Pearson, and F. Allgower. Online outlier detection and removal. In *MED99*, 1999.

[90] E. Minkov, R. C. Wang, A. Tomasic, and W. W. Cohen. Ner systems that suit user's preferences: adjusting the recall-precision trade-off for entity extraction. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 93–96. ACL, 2006.

[91] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on*

Natural Language Processing of the AFNLP: Volume 2-Volume 2, pages 1003–1011. ACL, 2009.

[92] R. Mitkov. *Anaphora resolution*. Routledge, 2014.

[93] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, January 2007.

[94] E. F. Nakamura, A. A. Loureiro, and A. C. Frery. Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Computing Surveys (CSUR)*, 39(3):9, 2007.

[95] G. Neubig, Y. Matsubayashi, M. Hagiwara, and K. Murakami. Safety information mining-what can nlp do in a disaster-. In *IJCNLP*, pages 965–973, 2011.

[96] S. Nunes, C. Ribeiro, and G. David. Using neighbors to date web documents. In *Proceedings of the 9th annual ACM international workshop on Web information and data management*, pages 129–136. ACM, 2007.

[97] M. Oita and P. Senellart. Deriving dynamics of web pages: A survey. In *TWAW (Temporal Workshop on Web Archiving)*, 2011.

[98] O. Okolloh. Ushahidi, or 'testimony': Web 2.0 tools for crowdsourcing crisis information. *Participatory learning and action*, 59(1):65–70, 2009.

[99] M. Palmer, D. Gildea, and N. Xue. Semantic role labeling. *Synthesis Lectures on Human Language Technologies*, 3(1), 2010.

[100] E. Pariser. *The filter bubble: What the Internet is hiding from you*. Penguin UK, 2011.

[101] T. Poibeau and L. Kosseim. Proper name extraction from non-journalistic texts. *Language and Computers*, 37(1):144–157, 2001.

[102] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

[103] M. F. Porter. Snowball: A language for stemming algorithms, 2001.

[104] Y. Qu, P. F. Wu, and X. Wang. Online Community Response to Major Disaster: A Study of Tianya Forum in the 2008 Sichuan Earthquake. In *Proceedings of the 42nd Hawaii International Conference on System Sciences*, 2009.

[105] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *VLDB 2001*, 2001.

[106] D. Ratkowsky. *Handbook of nonlinear regression models*. New York : M. Dekker, 1990.

[107] F. Reichartz, H. Korte, and G. Paass. Dependency tree kernels for relation extraction from natural language text. In *ECML PKDD '09: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 270–285, Berlin, Heidelberg, 2009. Springer-Verlag.

[108] S. Riedel and A. McCallum. Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 46–50. ACL, 2011.

[109] S. Riedel, D. McClosky, M. Surdeanu, A. McCallum, and C. D. Manning. Model combination for event extraction in bionlp 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 51–55. ACL, 2011.

[110] A. Ritter, O. Etzioni, S. Clark, et al. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM, 2012.

[111] A. Ross and A. Jain. Information fusion in biometrics. *Pattern recognition letters*, 24(13):2115–2125, 2003.

[112] K. Saito, R. J. Spence, C. Going, and M. Markus. Using high-resolution satellite images for post-earthquake building damage assessment: a study following the 26 january 2001 gujarat earthquake. *Earthquake spectra*, 20(1):145–169, 2004.

[113] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW2010*, 2010.

[114] H. M. SalahEldeen and M. L. Nelson. Carbon dating the web: estimating the age of web resources. In *TempWeb2013*, 2013.

[115] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11), 1975.

[116] S. Sarawagi. Information Extraction. *Found. Trends databases*, 1(3):261–377, Mar. 2008.

[117] J.-B. Sheu. An emergency logistics distribution approach for quick response to urgent relief demand in disasters. *Transportation Research Part E: Logistics and Transportation Review*, 43(6):687–709, 2007.

[118] S. Skakun, N. Kussul, A. Shelestov, and O. Kussul. Flood hazard and flood risk assessment using a time series of satellite images: A case study in namibia. *Risk Analysis*, 34(8):1521–1537, 2014.

[119] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine learning*, 34(1-3):233–272, 1999.

[120] R. Steinberger, A. Podavini, A. Balahur, G. Jacquet, H. Tanev, J. Linge, M. Atkinson, M. Chinosi, V. Zavarella, Y. Steiner, et al. Observing trends in automated multilingual media analysis. *arXiv preprint arXiv:1603.02604*, 2016.

[121] J. Strötgen and M. Gertz. HeidelTime: High quality rule-based extraction and normalization of temporal expressions. In *SemEval '10*, 2010.

[122] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[123] P. Talukdar, D. Wijaya, and T. Mitchell. Coupled temporal scoping of relational facts. In *WSDM'12*, 2012.

[124] X. Tannier. Extracting news web page creation time with dctfinder. In *LREC'14*, 2014.

[125] M. Thelwall. Extracting accurate and complete results from search engines: Case study windows live. *Journal of the American Society for Information Science and Technology*, 59(1):38–50, 2008.

[126] D. Tikk, P. Thomas, P. Palaga, J. Hakenberg, and U. Leser. A Comprehensive Benchmark of Kernel Methods to Extract Protein–Protein Interactions from Literature. *PLoS computational biology*, 6(7):e1000837, 2010.

[127] D. Tikk, P. Thomas, P. Palaga, J. Hakenberg, and U. Leser. A Comprehensive Benchmark of Kernel Methods to Extract Protein–Protein Interactions from Literature. *PLoS Comput Biol*, 6(7), 2010.

[128] H. Uszkoreit. Learning relation extraction grammars with minimal human intervention: strategy, results, insights and plans. In *Computational Linguistics and Intelligent Text Processing*, pages 106–126. Springer, 2011.

[129] F. B. Viégas, M. Wattenberg, and K. Dave. Studying cooperation and conflict between authors with history flow visualizations. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 575–582. ACM, 2004.

[130] S. Vieweg, A. L. Hughes, K. Starbird, and L. Palen. Microblogging during two natural hazards events: what twitter may contribute to situational awareness. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1079–1088. ACM, 2010.

[131] M. Viviani and G. Pasi. A multi-criteria decision making approach for the assessment of information credibility in social media. In *International Workshop on Fuzzy Logic and Applications*, pages 197–207. Springer, 2016.

[132] X. Wang, L. Tokarchuk, F. Cuadrado, and S. Poslad. Exploiting hashtags for adaptive microblog crawling. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 311–315. ACM, 2013.

[133] Y. Wang, B. Yang, et al. Harvesting facts from textual web sources by constrained label propagation. In *CIKM'11*, 2011.

[134] Y. Watanabe, Y. Okada, et al. Aligning articles in tv newscasts and newspapers. In *ACL '98*, 1998.

[135] M. Wick, A. Culotta, and A. McCallum. Learning field compatibilities to extract database records from unstructured text. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 603–611. ACL, 2006.

[136] M. Wyss. Real-time prediction of earthquake casualties. In *Disasters and Society – From Hazard Assessment to Risk Reduction*, pages 165–173, 2004.

[137] F. Xu, H. Uszkoreit, S. Krause, and H. Li. Boosting relation extraction with limited closed-world knowledge. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1354–1362. ACL, 2010.

[138] F. Xu, H. Uszkoreit, and H. Li. A Seed-driven Bottom-up Machine Learning Framework for Extracting Relations of Various Complexity. *ACL*, 7:584–591, 2007.

[139] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 4–11. ACM, 1996.

[140] E. Zangerle, W. Gassler, and G. Specht. Recommending#-tags in twitter. In *Proceedings of the Workshop on Semantic Adaptive Social Web (SASWeb 2011). CEUR Workshop Proceedings*, volume 730, pages 67–78, 2011.

[141] P. Zerfos, J. Cho, and A. Ntoulas. Downloading textual hidden web content through keyword queries. In *Digital Libraries, 2005. JCDL'05. Proceedings of the 5th ACM/IEEE-CS Joint Conference on*, pages 100–109. IEEE, 2005.

[142] C. Zhang, X. Zhang, W. Jiang, Q. Shen, and S. Zhang. Rule-based extraction of spatial relations in natural language text. *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, 2009.

[143] Z. Zhang, B. He, and K. C.-C. Chang. Understanding web query interfaces: Best-effort parsing with hidden syntax. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 107–118. ACM, 2004.

[144] G. Zhou and J. Su. Named entity recognition using an HMM-based chunk tagger. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 473–480, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

Selbstständigkeitserklärung

Ich erkläre, dass ich die Dissertation selbstständig und nur unter Verwendung der von mir gemäß § 7 Abs. 3 der Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät, veröffentlicht im Amtlichen Mitteilungsblatt der Humboldt-Universität zu Berlin Nr. 126/2014 am 18.11.2014, angegebenen Hilfsmittel angefertigt habe.

.....
Datum, Unterschrift